The GEDCOM 5.5.5 Specification with Annotations

Editor
Tamura Jones

Technical Reviewers

Bob Coret	creator of Genealogy Online and Open Archives
Diedrich Hesmer	creator of Our Family Book and GEDCOM Service Programs
Andrew Hoyle	creator of Chronoplex My Family Tree and the Chronoplex GEDCOM Validator
Kari Kujansuu	software developer for The Genealogy Society of Finland's Isotammi.net
Louis Kessler	creator of Behold, GEDCOM File Finder, and Double Match Triangulator
Stanley Mitchell	creator of ezGED Viewer
Nigel Munro Parker	creator of the GED-inline GEDCOM validator

First Release
2 Oct 2019

2019-10-05 edit: minor typo & link fixes.

The latest versions of the GEDCOM specification are available for download on www.gedcom.org.

Copyright

This publication, *The GEDCOM 5.5.5 Specification with Annotations* is based on the *The GEDCOM 5.5.1 Specification, Annotated Edition*, which is an annotated edition of the *GEDCOM 5.5.1 Specification*, which is a minor update of the *GEDCOM 5.5 Specification*. The GEDCOM 5.5 and 5.5.1 specifications were created by FamilySearch.

The *FamilySearch GEDCOM 5.5 specification* contains the following copyright notice:

> Copyright © 1987, 1989, 1992, 1993, 1995 by The Church of Jesus Christ of Latter-day Saints. This document may be copied for purposes of review or programming of genealogical software, provided this notice is included. All other rights reserved.

Similar copyright notices are included in other versions of the GEDCOM specification.

The *FamilySearch GEDCOM 5.5.1 specification* is largely identical to the *FamilySearch GEDCOM 5.5 specification*, yet contains a slightly different copyright notice:

> Copyright © 1987, 1989, 1992, 1993, 1995, 1999 by The Church of Jesus Christ of Latter-day Saints. This document may be copied for purposes of review only. It must not be used for programming of genealogical software while in draft, All other rights reserved.

The *FamilySearch GEDCOM 5.5.1 specification* isn't a draft, in fact never was a draft. The *FamilySearch GEDCOM 5.5.1 specification* is a publicly distributed specification, and it's the GEDCOM version that FamilySearch themselves use since their release of PAF 5.0 in 2000 CE. When the creator of a standard uses that standard in its major products, including a major public product, that standard isn't a draft.

This publication, *The GEDCOM 5.5.5 Specification with Annotations*, is published for the purposes of review and programming of genealogical software. It is based on the *The GEDCOM 5.5.1 Specification, Annotated Edition*, an in-depth and in-place review (for the purposes of programming), of the *FamilySearch GEDCOM 5.5.1 specification*. This is not only allowed by the GEDCOM 5.5 copyright notice, it is even allowed by the less permissive GEDCOM 5.5.1 copyright notice.

This publication is copyrighted too. The copyright notice is on the front page.

TABLE OF CONTENTS

Conventions ..... p. 4
About GEDCOM 5.5.5 ..... p. 6
What's New in 5.5.5 ..... p. 8
GEDCOM Writer & Reader Checklists ..... p. 26
Support for Earlier GEDCOM Versions ..... p. 28
Introduction ..... p. 29
    Purpose and Content of *The GEDCOM Standard* ..... p. 29
Chapter 1 Part I: GEDCOM grammar ..... p. 30
    Concepts ..... p. 30
    Grammar ..... p. 30
    Description of Grammar Components ..... p. 36
    White Space ..... p. 40
    CONC & CONT ..... p. 41
    Symbols Used with Basic GEDCOM Language and GEDCOM forms ..... p. 44
Chapter 1 Part II: Basic GEDCOM Language ..... p. 46
    Basic GEDCOM ..... p. 46
    GEDCOM Record Definitions ..... p. 47
    GEDCOM Tag Definitions ..... p. 51
Chapter 2: Lineage-Linked Form ..... p. 53
    Lineage-Linked Form Definition ..... p. 55
    Top-Level Records of the Lineage-Linked Form ..... p. 56
    Subrecords of the Lineage-Linked Form ..... p. 64
    Primitive Elements of the Lineage-Linked Form ..... p. 75
    The GEDCOM File ..... p. 109
    Sample Lineage-Linked GEDCOM File ..... p. 110
Chapter 3: Using Character Sets in GEDCOM ..... p. 112
    ANSEL ..... p. 113
    ASCII ..... p. 114
    Unicode ..... p. 114
Chapter 4: GEDCOM System Identifiers
    System Identifier Registration ..... p. 117
Appendix A: Lineage-Linked GEDCOM Tag Definition ..... p. 121
Appendix B: ANSEL To Unicode Conversion
Appendix C: ANSEL Character Set ..... p. 132
    Non-spacing graphic characters ..... p. 134
    Spacing graphic characters ..... p. 135

Bonus Chapters ..... p. 137
    GEDCOM Validation ..... p. 138
    GEDCOM Version Detection ..... p. 140
    GEDCOM 5.5.1 Version Detection ..... p. 142
    Event GEDCOM ..... p. 147
    Known GEDCOM Form Errors ..... p. 148
    more than one <<SUBMITTER_RECORD>> ..... p. 149
    No Standard for Multimedia File Transfer ..... p. 150
    GEDCOM 5.5 <<MULTIMEDIA_RECORD>> ..... p. 151
    GEDCOM 5.5.1 <MULTIMEDIA_LINK> ..... p. 153
    one- and two-digit years illegal ..... p. 157
    alias ALIA ..... p. 159
    NOTE.SOUR.NOTE.SOUR... ..... p. 160
    Maximum Path Length ..... p. 161
    The ANSEL Header demand ..... p. 162
    HEAD.CHAR.VERS ..... p. 164

Conventions

Language

This document uses English spelling and *logical quoting*; quotes go around an item, full stops are neither moved into nor out of quoted text.

must, must not, should, should not

This specification uses *must*, *must not*, *may*, *should* and *should not* as defined in IETF RFC 2119. Thus, other sentences in this specification almost always uses *must not* instead of *may not*. This specification sometimes italicises these phrases for emphasis, but does not capitalise these phrases.

Records

A GEDCOM files consists of *records*. A record that is contained within another record is a *subrecord*.
Previous versions of GEDCOM have also referred to records and subrecords as *structures* and *substructures*, even “record structures”. They've also been referred to as “tag”, “tag context” and just “context”, while “context” is generally used to refer to the enclosing record, not the subrecord itself.

A GEDCOM record consists of several parts, the *tag* is only part of the record. This version of GEDCOM consciously avoids using “tag” when “record” is meant.
A GEDCOM record starts with a level number. Records with level number zero are known as *top-level records*. Previous GEDCOM versions have referred to top-level records as “zero-level records”, “logical GEDCOM records”, “logical records”, and “record at level zero”; this specification always uses “top-level record”.

The term *records* includes *subrecords*, unless restricted by a modifier, as in “top-level records”, or “INDI records”.

Subrecord Notation

The GEDCOM specification uses a convenient subrecord notation to indicate particular subrecords. For example, INDI.NAME is the NAME subrecord of the INDI record, and HEAD.GEDC.VERS is the VERS subrecord of the GEDC subrecord of the HEAD record.

Bolded Tags

Tags in GEDCOM syntax definitions are bolded when they are mandatory in that context.

Characters, Code Units, and Code Points

Characters, code points and code units are related but different concepts. This specification assumes that the reader is familiar with these.

Previous GEDCOM versions expressed fields length in characters. GEDCOM 5.5.5 expresses field length in *code units*.

The text distinguishes between character sets and encodings; GEDCOM 5.5.5 demands the use of Unicode character set, either the UTF-8 or UTF-16 encoding. GEDCOM 5.5.5 continues the use of the HEAD.CHAR line value “UNICODE” (all-uppercase) to indicate UTF-16.

Terminator versus Newline

This text continues to use “terminator” for the GEDCOM line terminator, but uses “newline” for line breaks in (long) logical line values. This makes discussion of the CONT record easier to understand.

GEDCOM Readers & Writers

A *GEDCOM writer* is code that writes a particular version (and encoding) of GEDCOM. A *GEDCOM reader* is code that reads a particular version (and encoding) of GEDCOM.
A *GEDCOM 5.5.1 writer* is code that writes GEDCOM 5.5.1 files. A *GEDCOM 5.5.5 reader* is code that reads GEDCOM 5.5.5 files.
A typical genealogical system contains multiple GEDCOM readers and writers, for the last few versions of GEDCOM.

GEDCOM Examples

The GEDCOM specification contains some GEDCOM examples.
An effort has been made to ensure that all examples are valid GEDCOM, that do not only follow GEDCOM rules and best practice, but genealogical best practices as well.

Deprecated Features

The GEDCOM 5.5.5 specification contains few deprecated features.

Deprecated sections are highlighted through text & background colour, like this example:

escape_text:=

[alphanum | escape_text + alphanum | escape_text + space]

where:

space = U+0020, the Space character

The escape_text is the part of an escape between the opening and closing at sign (@). Escape sequences are defined by a GEDCOM form for that GEDCOM form.

GEDCOM 5.5.1 allows almost any Unicode character inside escape sequences.
GEDCOM 5.5.5 restricts escape sequences to alphanumerical characters and the space character.
Notice that escape_text must start with an alphanumerical character, that it must not start with a space.

The space character is allowed but deprecated; it is only included to keep the already existing escape sequence `@#DFRENCH R@` legal.

The general guideline for deprecated features is that they should not be used. Applications should avoid using deprecated features in the GEDCOM files they export.
The general guideline for import of deprecated features is that developers *should* not add support for deprecated features, but should not remove any already existing support either.

About GEDCOM 5.5.5

Maintenance Release

Technically, the GEDCOM 5.5.5 specification succeeds the GEDCOM 5.5.1 specification (1999 CE) as the current GEDCOM specification. Practically, it succeeds the *GEDCOM 5.5.1 Specification Annotated Edition* (2019 CE).

Previous GEDCOM releases were feature releases that introduced major new features. GEDCOM 5.5.5 does not introduce any major new features, quite the opposite: GEDCOM 5.5.5 removes long obsolete and deprecated features. GEDCOM 5.5.5 is a maintenance release; it fixes a variety of issues to provide a better specification, and a solid basis for future versions.

The *GEDCOM 5.5.5 Specification with Annotations* is based on the *GEDCOM 5.5.1 Specification Annotated Edition*. It is, despite extensive editing, essentially the same specification, but many textual corrections applied instead of highlighted, obsolete and deprecated parts removed, ambiguities and contradictions resolved. Both the GEDCOM grammar and the Lineage-Linked Form have been simplified without really changing things; whatever isn't allowed anymore wasn't being used anyway, or was deprecated decades ago.

The GEDCOM 5.5.5 focus is on corrections, clarifications, fixes and simplifications, but it's been twenty years since GEDCOM 5.5.1, and plenty of issues remain. The GEDCOM 5.5.5 specification is the GEDCOM 5.5.5 specification *with Annotations*. That isn't the ideal approach, it is the practical approach.

The GEDCOM 5.5.5 is based on the *GEDCOM 5.5.1 Annotated Edition* and contains some of the same annotations as the *Annotated Edition*.
Many annotations have been applied, replaced by a better specification, replaced by new sections or turned into bonus chapters, but yet other annotations remain.
The annotations and bonus chapter provide useful and interesting information, but are *not* part of the specification proper.

Quality

The GEDCOM 5.5.5 specification is about GEDCOM quality. GEDCOM 5.5.5 is a better GEDCOM; The GEDCOM 5.5.5 specification is a better specification, and GEDCOM 5.5.5 files are better GEDCOM files. The GEDCOM 5.5.1 and 5.5.5 specification are so close, that most high-quality GEDCOM 5.5.1 files are GEDCOM 5.5.5 files in all but version number. The GEDCOM 5.5.5 version number is badge of quality.

Developers of products that already produce GEDCOM 5.5.1 files that already conform with all GEDCOM rules, recommendations and best practices documented in the *GEDCOM 5.5.1 Annotated Edition* (2018 CE) will find it easy to upgrade the GEDCOM output of their products to GEDCOM 5.5.5.

GEDCOM 5.5.5 simplifies and tightens the rules to spark a change in genealogy industry culture. There are technical differences between GEDCOM 5.5.1 and GEDCOM 5.5.5, but the most important change is that GEDCOM 5.5.5 demands strict compliance.
Many existing GEDCOM 5.5.1 readers tolerate a wide variety of errors known to occur in GEDCOM 5.5.1 files,. GEDCOM 5.5.5 readers *need not* accepts errors known to occur in GEDCOM 5.5.1 files, and *must not* do so. GEDCOM 5.5.5 readers must make a clean break with such practices. Developers improving the GEDCOM output of their products should continue to use version number 5.5.1 until their GEDCOM output is in compliance with GEDCOM 5.5.5.

It is perfectly possible to combine a GEDCOM 5.5 and GEDCOM 5.5.1 reader into a single GEDCOM 5.5.0/1 reader, it is not possible to support GEDCOM 5.5.1 and 5.5.5 with a single combined reader.
A GEDCOM 5.5.5 reader must be separate from readers for GEDCOM 5.5.1 and earlier, not just to make sure it doesn't support obsolete Lineage-Linked Form records, but because the GEDCOM 5.5.5 grammar is considerably simpler and stricter than the GEDCOM 5.5.0/1 grammar. A GEDCOM 5.5.5 reader may use some of the same building blocks as the GEDCOM

5.5.1 reader, but it *can not* and *must not* be a combined GEDCOM 5.5.1 & 5.5.5 reader.
The GEDCOM 5.5.5 reader *need not* and *must not* compensate for errors that are only known to occur in ostensible GEDCOM 5.5.1 and earlier files, and *must* keep things simple by rejecting invalid files.
A GEDCOM 5.5.5 reader should be very clean code that is significantly simpler and noticeably faster than older GEDCOM readers.

Developers should keep their existing readers for GEDCOM 5.5.1 and earlier for backward compatibility, but are advised take advantage of the GEDCOM 5.5.5 specification to improve their GEDCOM 5.5.1 writer, and bring its output as close to that of the GEDCOM 5.5.5 writer as possible.

The *What's New in GEDCOM 5.5.5* section provides an overview of changes, including such things as CONC and CONT rules, tables of superfluous and obsolete records, and is followed by *GEDCOM writer & reader checklists*.

What's New in GEDCOM 5.5.5

This section provides an fairly detailed overview of changes between GEDCOM 5.5.1 and GEDCOM 5.5.5. The GEDCOM 5.5.5 specification does not include a overview of changes before GEDCOM 5.5.1. Those changes happened more than two decades ago, and are listed in the *GEDCOM 5.5.1 Annotated Edition*.

GEDCOM 5.5.5 isn't a feature release, it's a maintenance release; while a few small additions have been sneaked in, but there are no major new features.

Both the GEDCOM Grammar and the Lineage-Linked Form have been simplified. A major improvement is the introduction of a Basic GEDCOM Language and the top-level <GEDCOM_FILE> definition.
Text has been improved with current and more accurate terminology. When there were two ways to do the same thing, there's one way to do that thing now. Contradictions have been resolved, existing specifications has been clarified. Obsolete, deprecated and superfluous stuff has been taken out.

General Improvements

New Specification Parts

- A short *Conventions* section introduces some terminology and terminology changes.
- A section on CONC & CONT has been added to Chapter 1
- That new section on CONC & CONT is preceded by a new section about white space
- New sections defining the Basic GEDCOM Language
- The chapter on Character Set & Encodings has been rewritten.
- New chapter on system identifiers.
- GEDCOM Reader & Writer Checklists
- New appendix offers consolidated ANSEL / Unicode conversion tables (was bonus chapter in the Annotated Edition)
- Bonus chapters on a variety of issues, including GEDCOM Validation, GEDCOM version detection, and GEDCOM 5.5.1 version detection. The new bonus chapter on GEDCOM Version detection contains some advice on how to support GEDCOM 5.5.5 while continuing to support GEDCOM 5.5.1.

Textual Corrections

Corrections

Textual improvements include corrections of spelling errors, editing errors and conceptual errors, and improved terminology.
There are many small changes, some more important than others, but the many small changes really all add up to a significantly clearer, more consistent and more accurate specification.

English

As this is an international standard, an effort was made to use International English throughout.

Logical Quoting

Switched from illogical to logical quoting style to avoid confusion and misinterpretation; logical quoting style leaves no doubt what's quoted and what's not.

Terminology

The specification now uses better, more current, more accurate and more consistent terminology throughout. The text now uses “file” instead of “transmission”, “transcription” instead of “variation” and properly distinguishes between “tag” and “record” by no longer using “tag” when “record” is meant.
There is no more conflation of tag name and meaning into a mixed-case word such as “INDIvidual” either.

old	new
zero level record	top-level record
logical record	top-level record
tag	record
tag context	record
context	record
family record	family group record
transmission	file
variation	transcription
reading	transcription
biological	official
DESTination	DEST (destination)
MARRiage	MARR (relationship)
UNICODE	Unicode
UNICODE	UTF-16
characters	code units
sent the data	created the file
8-bit ASCII	(contradiction in terms)
ASCII (USA Version)	ASCII
record structures	records
(wide) characters	code units

Official versus Biological

The text of the specification no longer assumes that recorded relationship are biological. Relationships recorded on the basis of official paper documentation are official, and should not be assumed to be biological.

Family versus Family Group

The text is no longer written as if *family* and *family group* are the same concept. A so-called *family group* is *one couple and their children*. A single family often involves more than one such group.

Better Record Names

The FAM record is no longer referred to as the “family record”, but more accurately as the “family group record”. Similarly, the MARR record, which is not only used to record marriages, but for recording every relationship type between two people, is no longer referred to as the marriage record, but as the relationship record.

Appendix A Definitions

Significantly improved Appendix A definitions; many definitions are more objective, clearer and more accurate now.

Significant Simplification

One file per GEDCOM File

Multi-volume GEDCOM files (defined in GEDCOM 5.5.1 chapter 2) are no longer legal; a GEDCOM file must be just one file now. The multi-volume feature was obsolete before it was introduced, and remains largely unsupported. With GEDCOM 5.5.5 it is one file per GEDCOM file.

One GEDCOM Version per File

The GEDCOM 5.5.1 specification contains the demand to support a particular GEDCOM 5.5 record, while that record is not legal GEDCOM 5.5.1. The GEDCOM 5.5.1 <<MULTIMEDIA_LINK>> definition demands that applications support the GEDCOM 5.5 <<MULTIMEDIA_LINK>> record as well, because “some applications” use GEDCOM 5.5.1, yet the GEDCOM 5.5 <<MULTIMEDIA_LINK>> record.

GEDCOM does not demand anything like that. On contrary, GEDCOM 5.5.5 does not allow it; a GEDCOM form cannot make such a demand. From GEDCOM 5.5.5 onwards, it is one GEDCOM version per file; a GEDCOM 5.5.5 file must contain GEDCOM 5.5.5 records. A GEDCOM 5.5.5 reader must not tolerate anything that's only legal in another GEDCOM version.

One line terminator per GEDCOM file

The GEDCOM 5.5.5 definition of the line terminator makes is clear that the line terminator that ends each GEDCOM line must be the same for all GEDCOM lines in a GEDCOM file.

One Encoding per GEDCOM File

GEDCOM 5.4, 5.5 and 5.5.1 contains an *ANSEL demand*; the encoding of the GEDCOM header must use ANSEL, regardless of the encoding that header specifies for the (rest of) the GEDCOM file GEDCOM 5.5.5 demands that the encoding specified in the GEDCOM header be used for the entire file, including the header itself.

One Character Set for GEDCOM

GEDCOM 5.5.5 no longer allows ASCII or ANSEL. GEDCOM 5.5.5 is Unicode-only.

One Way of Doing Things

Where GEDCOM 5.5.1 still offers two ways of doing things, GEDCOM 5.5.5 offers one way. These changes have been a long time coming; GEDCOM 5.5.5 is only finishing transitions that FamilySearch started in GEDCOM 5.4 and 5.5 (1995 CE). Developers and users have had more than a quarter century to adopt the changes made by FamilySearch.

FamilySearch added structured addresses and deprecated unstructured addresses in GEDCOM 5.5. GEDCOM 5.5.5 demands structured addresses only.
Similarly, FamilySearch introduced the citation-source-repository model in GEDCOM 5.4, and deprecated the unstructured free-text citations. FamilySearch even gave instructions for providing structure in the previously free-form text field. GEDCOM 5.5.5 demands the use of source and repository records.

GEDCOM Grammar

There are clarifications, corrections, improvements, simplifications and more.

One file per GEDCOM file

The multi-volume support still demanded by GEDCOM 5.5.1 is a significant burden on GEDCOM readers; GEDCOM 5.5.1 readers must contain file handling logic to find and load multiple volumes from removable media, and prompt the user for a media change when a volume is missing. Its removal is a significant simplification of GEDCOM.

Text Files

The specification now explicitly states that GEDCOM files are text files, and that while GEDCOM allows multiple line terminators, each GEDCOM file must use a single line terminator choice throughout the file.

CR, LF & CR/LF

GEDCOM 5.5.5 continues to allow CR, LF & CR/LF as line terminators, but no longer allows LF/CR as a line terminator. This is a correction of a mistake; no platform uses LF/CR, and its inclusion creates problems for GEDCOM readers.
This correction is a real change to the GEDCOM grammar, yet does not affect any users; the LF/CR terminator only occurs in GEDCOM 5.5 and 5.5.1 test files specifically created to test support for it...

Tags Case-Sensitive

This version of the specification clearly states that GEDCOM tags are case-sensitive.

Legal Characters

The GEDCOM grammar has been edited to clearly separate which characters are legal in tags, identifiers and escapes sequences, and which characters are legal as line values.

Tags Underscore

Tags remain restricted to alphanumerical characters and the underscore. The tag definition has been tightened; the underscore may only occur as the first character now, and may no longer be the only character.
Incidentally, it was and remains legal for a tag to start with a digit.

Identifiers must be Alphanumerical

The GEDCOM 5.5.1 pointer definition allowed many characters inside cross-reference identifiers, even spaces, something that is almost sure to trip up a parser. It even allowed the bulk of all Unicode characters, thus creating more issues for a GEDCOM parser to deal with. In the real world, the flexibility provided by this definition was only used in GEDCOM files specifically created to test the ability of GEDCOM readers to handle this flexibility.

The GEDCOM 5.5.5 specification restricts the cross-reference identifier definition to alphanumerical characters, to allow straightforward parsing.
This allows GEDCOM 5.5.5 readers to be significantly simpler than GEDCOM 5.5.1 readers.

Escape Sequences

The updated GEDCOM grammar no longer allows almost any character in escape sequences. Escape sequences are restricted to alphanumerical characters, now except that spaces are still allowed. Spaces are allowed but deprecated because the Lineage-Linked Grammar defines one escape sequence with a space in it.

Tab is Legal

Use of Horizontal Tab in user text is legal now. Most genealogy applications already allowed the use of tabs in text, despite the GEDCOM 5.5.1 specifications disallowing it. GEDCOM 5.5.5 makes the existing practice legal.

Significant Tag Length

The GEDCOM grammar allows tags to be 31 code units long. The GEDCOM 5.5.1 specification adds that the first 15 of those 31 must be unique. What this really means is that while tags may be longer, their so-called *significant length* is only 15 code units. The practical consequence is that a GEDCOM reader may ignore everything beyond the 15th code unit, and that may result in two different identifiers being interpreted as being the same one.
This rule has never been off much practical consequence as all the standard GEDCOM tags as well as those of the Lineage-Linked Form are considerably shorter. Thus, this rule only affect GEDCOM extensions.
Nevertheless, GEDCOM 5.5.5 ends the complexity created by the limited significant length rule: the entire tag is significant now.

Maximum Record Size

The GEDCOM grammar in GEDCOM 5.5.1 limits the size of top-level records; they should fit in a memory buffer of less than 32K. That restriction made sense in the 1980s, it is obsolete now.
The GEDCOM grammar in the GEDCOM 5.5.5 specification no longer limits the size of top-level GEDCOM records.
Some records within the Lineage-Linked Form, which did not have an explicit maximum length before, have had their maximum length set at 32.765 code units.

line_item Fixed

The FamilySearch GEDCOM 5.5.1 specification defined line_item like this:

```
[ any_char | escape | line_item + any_char | line_item + escape ]
```

According to that recursive definition:

- line_item may consist of nothing but a single escape sequence.
- line_item may consist of a some text followed by an escape sequence.
- line_item may consist of a some text followed by multiple escape sequence.
- line_item may consist of a some text both preceded and followed by an escape sequence.
- line_item may consist of nothing but multiple escape sequences.

Most of that is not as intended. Most of these possibilities are wrong.

The GEDCOM 5.5.5 line_item definition explicitly restricts line_item to just three different formats:

- just text
- just one escape sequence
- just one escape sequence followed by text

pointer is an xref_ID

GEDCOM 5.5.1 defines xref_ID as a pointer. That's the wrong way round. GEDCOM 5.5.5 defines a pointer as an xref_ID. As part of this change, pointer_string has been renamed identifier_string.

White Space Rules

The new *White Space* section adds white space rules:

- **GEDCOM lines *must not* not end with superfluous trailing white space.**
- **GEDCOM lines *may* contain significant trailing white space.**
- **A GEDCOM reader *must* import every line value as-is, *must not* not strip trailing white space.**
- **A GEDCOM reader *must* import every line value as-is, *must not* not strip leading white space.**

New CONC & CONT Rule, Clarification and Guideline

- Rule: The CONC & CONT records *must not* be used with records defined in the GEDCOM grammar. They may only be used with records defined in a GEDCOM form.
- Clarification: It is illegal to have CONC or CONT records subordinate to CONC or CONT records.
- Guideline: GEDCOM writers *should not* use CONC records for line values with a maximum length of 248 code unit or less.

Simple CONC & CONT Processing Rules

The new section on CONC & CONT ends with CONC & CONT processing rules for GEDCOM writers & readers to ensure correct transfer of white space. Those rules are the simplest yet, literally *requiring no effort* from GEDCOM readers at all; a GEDCOM reader must import all lines *as-is*.

Character Sets & Encodings

Unicode

Unicode replaces ANSEL as *the* GEDCOM character set. GEDCOM 5.5.5 does no longer allow ASCII or ANSEL. GEDCOM 5.5.5 demands Unicode. The recommended encoding is UTF-8+ for Western applications, UTF-16 for Eastern applications. Applications, when asked to create a GEDCOM file, need not offer users a choice between these two.

Import of either encoding should never be an issue. Applications *must* be able to import both UTF-8 and UTF-16 GEDCOM 5.5.5 files.
Adding UTF-16 GEDCOM import to a Unicode application that already support UTF-8 GEDCOM import is fairly easy to do, as the application is using UTF-16 internally. The import only needs to pay attention to the normal form, and the application can rely on operating system routines for that.

Unicode Applications Only

GEDCOM 5.5.5 is for Unicode applications. Legacy applications that are still code page-based *must not* use GEDCOM 5.5.5, but must continue to use GEDCOM 5.5.1. Code page applications can and should follow all the rules, recommendations and best practices provided the *GEDCOM 5.5.1 Annotated Edition*.

Byte Order Mark Mandatory

Unicode demands Byte Order Mark on UTF-16 files, allows them on UTF-8 files. GEDCOM 5.5.5 keeps it simple: use of a Byte Order Mark (BOM) on UTF-8 GEDCOM files is mandatory for all GEDCOM 5.5.5 files, both UTF-8 and UTF-16 GEDCOM files. This simplifies GEDCOM detection considerably. Systems reading GEDCOM 5.5.5 files know the character set and encoding used in the header as soon as they've processed the Byte Order Mark (BOM), *before*

they start reading the header.

Unicode Code Point Notation

The specification now uses Unicode point notation to indicate specific characters; e.g. U+0020 is the Unicode code point for the Space character.

ANSEL

Starting with GEDCOM 5.5.5, the ASCII and ANSEL character sets are illegal. It is illegal to use ANSEL in GEDCOM 5.5.5, so a system need not support ANSEL at all to be fully GEDCOM 5.5.5-compliant. Systems should support ANSEL for import of ANSEL-encoded GEDCOM 5.5.1 and earlier files.

Consolidated ANSEL Table

A single consolidated ANSEL table solves the problem of different “ANSEL” tables in different GEDCOM versions. This single consolidated table can and should be use for all older GEDCOM versions; that's why it's there.

Strict Conformance

GEDCOM 5.5.5 does not tolerate the use of illegal characters sets or encodings.

- GEDCOM 5.5.5 writers *must not* use anything but the legal encodings.
- GEDCOM 5.5.5 readers *must not* accept anything but the legal encodings.
- GEDCOM 5.5.5 readers *must* reject all illegal encodings.

GEDCOM Language

Basic GEDCOM Language

The GEDCOM 5.5.1 *Chapter 1 Data Representation Grammar* defines just two records types: CONC & CONT. The HEAD & TRLR record are not even mentioned.
The chapter has been extensively rewritten and extended to define the Basic GEDCOM Language, including the HEAD & TRLR records.

Multiple GEDCOM Forms Problem

The GEDCOM 5.5.1 specification allows multiple GEDCOM forms, yet it doesn't...
GEDCOM 5.5.1 Chapter 2 defines the Lineage-Linked Form, and that includes the definition of the GEDCOM header, which specifies the GEDCOM form being used. Specifying a GEDCOM form within the Lineage-Linked GEDCOM header does not make sense, as that entire GEDCOM header is part of the Lineage-Linked Form.
The HEAD.GEDC.FORM line value supposedly specifies the GEDCOM form, but `LINEAGE-LINKED` is the only legal value, and that suggests that the HEAD.GEDC.FORM record is superfluous and redundant.

Basic GEDCOM Language Solution

The GEDCOM 5.5.5 specification distinguishes between the Basic GEDCOM Language and the Lineage-Lineage Link Form. The record that specifies the GEDCOM form is no longer part of a specific GEDCOM form, but part of the Basic GEDCOM Language.
GEDCOM 5.5.5 defines a Basic GEDCOM Language first, and only then defines the Lineage-Linked Form. The Basic GEDCOM language defines the GEDCOM header and trailer record in addition to the CONC & CONT records. The HEAD.GEDC.FORM is part of the Basic

GEDCOM Language now, and really specifies which GEDCOM form is being used.

GEDCOM Header Split

The old GEDCOM header has been split into a basic GEDCOM and an extension. The Basic GEDCOM header is defined by the Basic GEDCOM Language, the extension is defined by the Lineage-Linked Form.

Trailer Record

In GEDCOM 5.5.1, the TRLR record signifies the end of the GEDCOM file.
However, in GEDCOM 5.4, the TRLR record may be followed by embedded multi-media objects, with the real end of the GEDCOM file marked by an the END record.
GEDCOM 5.5 still supports embedded multimedia records, but they appear before the TRLR record, just like all other records.

By making the header and trailer records part of the Basic GEDCOM language, and demanding that all GEDCOM form records occur in between, the GEDCOM 5.5.5 specification ensures that the trailer record is truly the trailer record.

CONC & CONT

CONC & CONT Section

The *GEDCOM 5.5.1 Annotated Edition* collected information and instructions about CONC & CONT together in a large annotation. The GEDCOM 5.5.5 specification provides a new *CONC & CONT* section as part of Chapter 1.

CONC & CONT Restriction

The GEDCOM specification allows CONC & CONT anywhere they are needed. GEDCOM 5.5.5 introduces a restriction, that should have been part of GEDCOM from the beginning, to keep parsing GEDCOM files simple; CONC & CONT must not be used with basic GEDCOM records, only with records defined by a GEDCOM form.

It is illegal to use CONC or CONT within the Basic GEDCOM header. It is legal, but strongly discommended, to use CONC or CONT within a form-specific header extension. It is important to keep the GEDCOM header simple. The GEDCOM header is not only examined by GEDCOM readers, but also by general utilities that report file types.

New CONC Guideline

GEDCOM 5.5.5 introduces a new guideline for CONC usage; GEDCOM writers should not use CONC for line values with a maximum length of 248 code units or less.
A GEDCOM reader must not complain about unnecessary use of CONC records as long as they're valid, but a GEDCOM validator should give a warning.

Simple CONC & CONT Rules

Historically, CONC & CONT handling has been problematic. The *GEDCOM 5.5.1 Annotated Edition* provided simple guidelines for getting it right.
The GEDCOM 5.5.5 specification provides a new section about white space, and then provides the simplest rules possible:

- A GEDCOM writer must not write superfluous white space, may only write significant white space.
- A GEDCOM reader must treat all white space as significant; it may never trim leading or

trailing white space from lines values.

Those two simple rules ensure correct transfer of long text files, even for pathological cases such as long stretches of white space.

GEDCOM Header

Simplified GEDCOM Detection

All GEDCOM 5.5.5 files are Unicode files, and all GEDCOM 5.5.5 files, both UTF-8 and UTF-16 files, start with a Byte Order Mark (BOM). The mandatory Byte Order Mark simplifies GEDCOM detection and header interpretation. Systems reading GEDCOM 5.5.5 files know the character set and encoding used in the header as soon as they've processed the Byte Order Mark (BOM), that is *before* they start reading the header.

Strict Conformance

GEDCOM 5.5.5 demands a correct GEDCOM header. A GEDCOM 5.5.5 reader *must not* process an ostensible GEDCOM file if the header is invalid.

GEDCOM Header Split

The GEDCOM header defined in GEDCOM 5.5.1 contains both general subrecords as well as subrecords specific to Lineage-Linked Form. To continue to accommodate both categories in GEDCOM 5.5.5 with just one GEDCOM header, the GEDCOM header has been split into a Basic GEDCOM header and a Lineage-Linked Form extension.
The Basic GEDCOM Language defines the Basic GEDCOM header and allows GEDCOM forms to extend it with form-specific records. The Lineage-Linked Form extends the GEDCOM header with Lineage-Linked specific subrecords.

GEDCOM Recognition

The definition of a Basic GEDCOM header moves all the essential header subrecords to the very front of the GEDCOM file. This aids recognition of GEDCOM files.

HEAD.GEDC.FORM.VERS

Historically, there is just one version number for both the Basic GEDCOM language and the Lineage-Linked Form.
GEDCOM 5.5.5 adds HEAD.GEDC.FORM.VERS; a version number for the GEDCOM form used, distinct from the GEDCOM version number.

CONC & CONT

That CONC & CONT must not be used with basic GEDCOM records, only with record defined by a GEDCOM form, specifically means that CONC & CONT must not be used in the <<GEDCOM_HEADER>>.
Use of CONC & CONT is allowed but strongly deprecated for header subrecords defined by a GEDCOM form.

HEAD.DATE.COPR and HEAD.NOTE

To avoid needing CONC & CONT with header subrecords of the Lineage-Linked Form, the maximum length of the HEAD.DATE.COPR and HEAD.NOTE lines values have both been set at 248 code units.
The 248 code units allowed for <COPYRIGHT_GEDCOM_FILE> is significantly more than the

maximum of 90 code units allowed by GEDCOM 5.4 and 5.5. The 248 code units allowed for <GEDCOM_CONTENT_DESCRIPTION> is more than the maximum 213 code units allowed by GEDCOM 5.4, and identical to the maximum of 248 code units allowed by GEDCOM 5.5.

GEDCOM Dialect

GEDCOM 5.5.5 clearly states that it is the HEAD.DEST line value (*not* the HEAD.SOUR line value) that specifies the GEDCOM dialect used, i.e. the interpretation of GEDCOM extensions.

HEAD.DEST Value Rules

- GEDCOM 5.5.5 clearly states that the HEAD.DEST line value should default to being equal to the HEAD.SOUR line value.
- GEDCOM 5.5.5 adds that it is illegal for the HEAD.DEST to be a nonsense value.

HEAD.SUBM

GEDCOM 5.5.5 allows just one SUBM record for the entire file, and it must come directly after the GEDCOM header. That makes the HEAD.SUBM pointer superfluous. The HEAD.SUBM record is no longer mandatory, and has been deprecated.

XREF.SUBM

Obviously, the XREF:SUBM pointer has been deprecated too.

Lineage-Linked Form

removal of at-signs

The many at-signs (@), which used to be so characteristic of Lineage-Linked Form definitions, are gone from those definitions. Lineage-linked form definitions used fragments such as `@<XREF:INDI>@`, always surrounding a cross-reference with the at-signs that start and end a cross-reference. The problem with that is not only that it is odd for the record using a cross-reference to demand at-signs around a cross-reference, but that the GEDCOM grammar makes it clear that *the at-signs are already part of the cross-reference*. Thus, those definitions were actually demanding double at-signs.

Renamed

Some parts of the Lineage-Linked Form have been renamed to ease understanding, avoid confusion and enhance readability:

- <<RECORD>> to <<LINEAGE_LINKED_RECORD>>
- <<FAM_RECORD>> to <<FAM_GROUP_RECORD>>
- <APPROVED_SYSTEM_IDENTIFIER> to <SYSTEM_IDENTIFIER>
- <CHARACTER_SET> to <CHARACTER_ENCODING>
- <FILE_NAME> to <GEDCOM_FILE_NAME>
- <TRANSMISSION_DATE> to <FILE_CREATION_DATE>
- <NAME_PHONETIC_VARIATION> to <NAME_PHONETIC>
- <PLACE_PHONETIC_VARIATION> to <PLACE_PHONETIC>
- <PHONETIC_TYPE> to <PHONETISATION_METHOD>
- <NAME_ROMANIZED_VARIATION> to <NAME_ROMANISED>
- <PLACE_ROMANIZED_VARIATION> to <PLACE_ROMANISED>
- <ROMANIZED_TYPE> to <ROMANISATION_METHOD>
- <NATIONAL_ID_NUMBER> to <ID_NUMBER>

- <SUBMITTER_TEXT> to <USER_TEXT>
- <COUNT_OF_MARRIAGES> to <NUMBER_OF_RELATIONSHIPS>

Maximum GEDCOM File Length

The maximum length of <GEDCOM_FILE_NAME.SUBM> has been increased from an arbitrary 90 to 248 code units - the maximum length a single GEDCOM line will allow.

NEW_TAG

There is no tag called NEW_TAG and instructions concerning user-defined record should not be hidden among the definition of GEDCOM form records. There is a small *User-Defined Records* section now.

Duplicate Records

The GEDCOM 5.5.1 Lineage-Linked From defines several records duplicate functionality of other already existing records. The GEDCOM 5.5.1 Annotated Edition (2018 CE) marked these records as deprecate. GEDCOM 5.5.5 simplifies GEDCOM by removing the duplicate records. The following table summarises which records were still legal in GEDCOM 5.5.1, and what should be used in GEDCOM 5.5.5 instead:

allowed in GEDCOM 5.5.1	GEDCOM 5.5.5
INDI.BAPL	INDI.BAPM
INDI.BLES	INDI.EVENT
INDI.CONL	INDI.CONF
INDI.ENDL	INDI.EVENT
INDI.ORDN	INDI.EVENT
INDI.SLGC	INDI.EVENT
INDI.SSN	INDI.IDNO
FAM.SLGS	FAM.EVENT

LDS Events

The above table contains LDS-specific records. Removal of the LDS-specific records improves support for these LDS events.
Few genealogy applications support the LDS-specific records. On GEDCOM export from those applications and subsequent import into another application, the data recorded in these events are lost. By requiring the use of the universally supported regular baptism record instead of a poorly supported LDS-specific baptism record, GEDCOM 5.5.5 ensures that the recorded information is no longer lost in transfer.

Obsolete Records

The following obsolete and superfluous GEDCOM 5.5.1 records have been retired, and do not have GEDCOM 5.5.5 alternatives.

allowed in GEDCOM 5.5.1	brief comment
HEAD.CHAR.VERS	Should never have been included. Never used, only abused.
HEAD.PLAC HEAD.PLAC.FORM	Unnecessary complexity. Widely unsupported.
HEAD.SUBN	Specific to Ancestral File; obsolete.
EVEN.RESN	Specific to Ancestral File; obsolete.
FAM.RESN	Specific to Ancestral File; obsolete.
FAMC.STAT	Never used, unsupported not needed.
FAM.SUBM	GEDCOM 5.5.5 has one submitter per GEDCOM file.
INDI.RESN	Specific to Ancestral File; obsolete.
INDI.SUBM	GEDCOM 5.5.5 has one submitter per GEDCOM file.
INDI.AFN	Specific to Ancestral File; obsolete can use IDNO to maintain but should not.
INDI.ALIA	Never used in practice. ASSO remains supported.
INDI.ANCI	Never used, not needed.
INDI.DESI	Never used, not needed.
INDI.RFN	Specific to Ancestral File; obsolete can use IDNO to maintain but should not.
SUBM.LANG	Serves no purpose. Already have HEAD.LANG.
SUBM.RFN	Specific to Ancestral File; obsolete.
SUBN	Specific to Ancestral File; obsolete.

<CHILD_LINKAGE_STATUS>

The <CHILD_LINKAGE_STATUS> is not supported by any major applications, and quite possibly by no application at all. It has been marked obsolete in the *GEDCOM 5.5.1 Annotated Edition*, and removed from GEDCOM 5.5.5.
As a result, the STAT tag is obsolete too, no longer used and has been removed from Appendix A.

Tags obsoleted in GEDCOM 5.5.5

The following tags, legal in GEDCOM 5.5.1, do not occur in GEDCOM 5.5.5 files: AFN, ALIA, ANCI, BAPL, CONL, DESI, ENDL, ORDN, SLGC, RESN, SSN, SLGS, SUBN, and STAT.

Maximum Identifier Length

Contradiction between the GEDCOM grammar and the Lineage-Linked Form concerning the maximum length of cross-reference identifiers has been resolved. The maximum length is as specified by the GEDCOM grammar.

C-style Comments

C-style comments in definitions and GEDCOM examples were not universally understood as comments. When they were recognised as comments, they created the mistaken impression that GEDCOM files may contain comments.
GEDCOM does not support comments. Removed all C-style comments from definitions.

Confusing CONC & CONT

The GEDCOM grammar makes it clear that CONC & CONT are allowed anywhere they are needed. The explicit inclusion of CONC & CONT in the Lineage-Linked Form is an unnecessary complication of the syntax, that encouraged misinterpretation, as explicit inclusion at a few select points in the syntax strongly suggests is not allowed elsewhere...
GEDCOM 5.5.5 respects that this is implicit in any GEDCOM from, and no longer confuses by including CONC or CONT explicitly. All explicit CONC & CONT references have been removed

from the Lineage-Linked Form definition. The definitions in the Lineage-Linked Form are all about the Lineage-Linked Form now, as they should be.

The GEDCOM 5.5.1 specification did not set maximum length for the records concerned. The GEDCOM 5.5.5 specification sets their maximum length at either 248, 4095 or 32.767 (2^15-1) code units. This is in deference to the old (GEDCOM 5.5.1) rule that GEDCOM records should fit into a memory of less than 32KB. A Unicode application using UTF-16 internally will need no more than a 64 KB buffer to hold the largest legal values.

item	maximum
IND.DSCR <PHYSICAL_DESCRIPTION>	4095
SOUR.AUTH <SOURCE_ORIGINATOR>	248
SOUR.TITL <SOURCE_DESCRIPTIVE_TITLE>	4095
SOUR.PUBL <SOURCE_PUBLICATION_FACTS>	4095
SOUR.DATA.TEXT <TEXT_FROM_SOURCE>	32.767
SOUR.TEXT <TEXT_FROM_SOURCE>	32.767
NOTE <USER_TEXT>	32.767

Same-Sex Marriage

GEDCOM 5.5.5 supports same-sex relationships. Technically, this is not a new feature, but merely a clarification in support of already existing practice. As the *Annotated Edition* already noted, the GEDCOM 5.5.1 specification does *not* really exclude same-sex marriage, but actually supports it *as-is*, i.e. without resorting to any GEDCOM extension. The Lineage-Linked Form specification for recording relationships in the GEDCOM 5.5.5 specification has not changed from GEDCOM 5.5.1. There's no change in syntax, only improved descriptions.

Sex specified by INDI.SEX

GEDCOM 5.5.5 clearly states the following two rules, already present in the Annotated Edition:

- No assumption about gender should be made based on the usage of the FAM.HUSB or FAM.WIFE records.
- An individual's sex is specified by INDI.SEX, and by INDI.SEX only.

relationships

The FAM.MARR record documents the relationship between FAM.HUSB or FAM.WIFE. The nature of the relationship is documented by the optional MARR.TYPE subrecord, and GEDCOM 5.5.5 provides a list of the common MARR.TYPE values and their meaning in the definition of the <<FAM_GROUP_RECORD>>.

value	description
unknown	relationship (type unknown)
marriage	marriage
not married	not married
civil	civil marriage
religious	religious marriage
common law	common law marriage
partnership	partnership
registered partnership	registered partnership
living together	living together
living apart together	living apart together

GEDCOM Examples

Leading White Space in Examples

GEDCOM does not allow GEDCOM lines to have any leading white space. The GEDCOM specification should lead by example; all examples within the specification should be valid examples. No more “do as we say, not as we do”. All leading white has been removed from the examples.

Best Practices

Some GEDCOM examples contained bad practice such as incomplete or abbreviated place names. The GEDCOM specification should promote best practices. The examples have been improved to use complete, unabbreviated place names.

Obsolete and Superfluous Records

Ancestral File

Ancestral File is a long retired system. All records in support of Ancestral File are obsolete, and have been removed. This includes the <SUBMISSION_RECORD> and the SUBM.RFN record.

SUBM.LANG

The SUBM.LANG record, which allowed specifying a language preferred by the submitter, does not serve a clear purpose. The GEDCOM header already provides HEAD.LANG to specify the language used in the GEDCOM file, and specifying a preferred language only makes sense if you expect a response. The SUBM.LANG field is superfluous at best, and not used in practice. The *GEDCOM 5.5.1 Annotated Edition* marks it as deprecated, GEDCOM 5.5.5 removes its.

<PLACE_HIERARCHY>

The intended complexity of the <PLACE_HIERARCHY> feature was insufficiently defined and remained widely unsupported. The <PLACE_HIERARCHY> feature has been removed. All place names must be specified as comma-and-space separated place parts, listed from smallest to largest jurisdiction.

<SOURCE_CITATION>

GEDCOM 5.5.1 already deprecated, but still allowed the old <SOURCE_CITATION> format in addition to the new one introduced in GEDCOM 5.4 (1995 CE). GEDCOM 5.5.5 simplifies source citations by allowing just the “new” source citation format, which includes a reference to a SOUR record.

<SOURCE_REPOSITORY_CITATION>

The GEDCOM 5.5.1 <SOURCE_REPOSITORY_CITATION> structure definition is really two alternate structure definitions that are presented as one. An odd side-effect of this confusing presentation is that, upon first reading, the <SOURCE_REPOSITORY_CITATION> structure definition seems to be the only record in the GEDCOM 5.5.1 specification that expect a pointer value, yet allows it to be absent (<NULL>). That is not the case. What is really going is that GEDCOM 5.5.1 supports source citations that do reference repositories and source citations that do not. The *GEDCOM 5.5.1 Annotated Edition* shows the split into the two different definitions. GEDCOM 5.5.5 simplifies <SOURCE_REPOSITORY_CITATION> (which is an *optional* subrecord of <<SOURCE_RECORD>>) by demanding that, if present, it references a repository, as its name suggest.

One Submitter per File

GEDCOM no longer allows individual records to labelled with submitters. This feature was not used in practice, because it has no practical usage, not even for shared trees. GEDCOM 5.5.5 keeps it simple by allowing only one submitter, and therefore only one <SUBMITTER_RECORD>, for the entire file.
This makes the HEAD.SUBM reference to that single submitter record superfluous, but it is kept for compatibility's sake.

One Multimedia Record per File, one File per Multimedia Record

GEDCOM 5.5.1 allows multiple media files per multimedia record. This is unnecessary, and remains poorly supported, so export to GEDCOM import into another system is likely to result in data loss.
GEDCOM 5.5.5 solves that data loss issue by simplifying multimedia support. GEDCOM 5.5.5 demands one multimedia record per file, one file per multimedia record.

SSN

GEDCOM version 5.3 through 5.5.1 define a SSN record for the U.S. Social Security Number, duplicating the already existing IDNO record functionality.
GEDCOM 5.5.5 uses the IDNO for all third-party numbers. Application can identify Social Security numbers with the IDNO.TYPE value “SSN”.

It is best practice for GEDCOM writers using GEDCOM 5.5.1 or earlier to ignore the SSN record, and always use the IDNO record, just like a GEDCOM 5.5.5 writer does.

Fixes

EMAIL not EMAI

The EMAIL record always uses an EMAIL tag. The confusing EMAI (no L) definition in Appendix A has been replaced with an EMAIL definition.

single <CHANGE_DATE>

Replaced two different <CHANGE_DATE> definitions (GEDCOM 5.5.1 page 31 and 44) with a single <CHANGE_DATE> definition

ASSO Definition in ASSO Definition

Moved ASSO definition and example from <INDIVIDUAL_RECORD> definition to <ASSO_RECORD> definition.

Maximum Length Multimedia Files Names

Fixed the maximum length of multimedia files names, from 30 to 259 code units.

DATE Spaces

Dates contain spaces. Fixed all the date definitions to include spaces in the syntax.

Hebrew B.C.

Removed the erroneous “B.C.” from the Hebrew date syntax.

Dual-Style Dates: Julian Calendar

England and its colonies used dual-style dates with the Julian Calendar. However, FamilySearch GEDCOM 5.4 through GEDCOM 5.5.1 restricts dual-style dates to the Gregorian Calendar... GEDCOM 5.5.5 allows dual-style Julian Calendar dates.

@#DROMAN@

The escape sequence `@#DROMAN@` has been removed from <DATE_CALENDAR_ESCAPE> because there is no associated calendar.

<PERSONAL_NAME_PIECES> Mandatory

To prevent unnecessary loss of information on export & import, the NPFX, GIVN, NICK, SPFX, SURN, and NSFX subrecords that identify name parts of the <PERSONAL_NAME_STRUCTURE> are no longer optional. Each of the subrecords remains optional, because no name parts occurs in every recorded name, but all name parts must be clearly identified.

<MULTIMEDIA_FORMAT> Fixes

Removed `OLE` from the list of multimedia formats: OLE is not a multimedia format at all, it is a technology. GEDCOM 5.5 uses the values `JPEG` and `TIFF`, GEDCOM 5.5.1 uses `JPG` and `TIF` instead.
GEDCOM 5.5.5 uses the values `JPEG` and `TIFF`, not `JPG` and `TIF`. The `JPG` and `TIF` values are supported, but deprecated.
The list of allowed multimedia formats has been expanded, but the fact that the enumeration of allowed formats is supposed to be exhaustive remains a fundamental issue.

<AGE_AT_EVENT> Centenarian Support

The <AGE_AT_EVENT> definition allowed only 2 digits for the year. <AGE_AT_EVENT> allows three digits for the year now.

TYPE subrecords in <<INDIVIDUAL_ATTRIBUTE_STRUCTURE>>

The GEDCOM 5.5.1 <<INDIVIDUAL_ATTRIBUTE_STRUCTURE>> description states that the FACT and IDNO records *require* a TYPE subrecords, while it *may* be used with all the other record types, while there is no TYPE record is not in the syntax! GEDCOM 5.5.5 adds the TYPE subrecords to the syntax.

<INDIVIDUAL_EVENT_STRUCTURE>

The updated <INDIVIDUAL_EVENT_STRUCTURE> syntax no longer allows `1 BIRT Y`; we already know that individuals were born; the only thing that record offers is a chance for a GEDCOM reader to get confused by it. The BIRT record should only be used to provide real information.

Clarifications

GEDCOM Form is Case-Sensitive

Within GEDCOM, so-called controlled line values (exhaustive enumerations) are case-insensitive, but the GEDCOM form *is not* a controlled line value, quite the opposite. The GEDCOM form value is case-sensitive. The line value used for the Lineage-Linked form, `LINEAGE-LINKED`, must be written all-uppercase.

Submitter after Header

The syntax of Lineage-Linked Form has been clarified to explicitly demand that the one mandatory <SUBMITTER_RECORD> directly follow the <HEADER_RECORD>.

Leading Zeroes on Day

Resolved open issue: <DAY>, the day of the month within a date, may *not* start with leading zeroes.

Proper Version Numbers

GEDCOM versions up through GEDCOM 5.5.1 (and the GEDCOM 5.6 draft) defined <VERSION_NUMBER> as an arbitrary string. GEDCOM 5.5.5 demands that the HEAD.SOUR.VERS and HEAD.GEDC.VERS line values be version numbers, and nothing else. The version number records can no longer be abused to contain the product name, system id, release status or platform. GEDCOM 5.5.5 readers will encounter nothing but a version number.

Two Version Number Formats

GEDCOM 5.5.1 replaces the single <VERSION_NUMBER> definition with two separate definitions; <GEDCOM_VERSION_NUMBER> and <PRODUCT_VERSION_NUMBER>. <GEDCOM_VERSION_NUMBER> demands a version number in the *major.minor.revision*format. <PRODUCT_VERSION_NUMBER> demands a version number in the *major.minor.revision.build* format.

Default <SEX_VALUE>

The default <SEX_VALUE> is `U` for *Unknown*.

<PEDIGREE_LINKAGE_TYPE>

Kept the *GEDCOM 5.5.1 Annotated Edition* remark that GEDCOM does not support DNA or biological relationships.
Added statement that, in absence of <PEDIGREE_LINKAGE_TYPE>, the value `birth` for official parents is assumed.

New Features

HEAD.GEDC.FORM.VERS

The version number for the GEDCOM form, separate from the GEDCOM version number.

New SEX Values

- Added `X`, meaning *Intersex* as a possible <SEX_VALUE>
- Added `N`, meaning *Not recorded* as a possible <SEX_VALUE>

Birthdays: Dates without Years

The <DATE_GREG> definition has been expanded to include exact dates without a year; this acknowledges that we often know a birt hday while being unsure of the birth year.

BCE

The *GEDCOM 5.5.1 Annotated Edition* already recommended that GEDCOM readers accept both `BC` and `BCE` in addition to `B.C.`. GEDCOM 5.5.5 makes the `BCE` and `BC` values official. All three values mean the same thing. The religiously neutral `BCE` is preferred, the other two values are deprecated. However, it is recommended that developers who want to maximise compatibility of their GEDCOM 5.5.1 and GEDCOM 5.5.5 output, use `B.C.` for now.

GEDCOM Writer & Reader Checklists

GEDCOM 5.5.5 enables simple, straightforward and much faster GEDCOM readers by demanding strict compliance from GEDCOM writers. GEDCOM 5.5.5 writers *must* produce correct GEDCOM, so that developers need not spend time complicating their GEDCOM 5.5.5 readers with support for all kinds of third-party errors. In fact, to really ease the burden, a GEDCOM 5.5.5 reader *must not* support erroneous import. This keeps GEDCOM 5.5.5 readers simple, and forces all developers to produce quality GEDCOM files. Developers that want to continue to produce invalid GEDCOM files, must keep using GEDCOM 5.5.1.

The following two checklists are offered as an aid to development.

GEDCOM 5.5.5 Writer

- *must* use the Unicode character set, using either the UTF-8 or UTF-16 encoding
- *must* make sure that it's UTF-8 output is really UTF-8, not CESU-8
- *must* start each GEDCOM file with Byte Order Mark (BOM)
- *must* use a proper system identifier
- *must* use a proper version number for its system
- *must* default HEAD.DEST to the same system identifier as HEAD.SOUR
- *must* specify GEDCOM version 5.5.5
- *need not* create a HEAD.SUBM record, but if it does, the pointer must be valid
- *should* specify GEDCOM form `LINEAGE-LINKED`, there is no other GEDCOM form
- *does not* output superfluous white space
- does not include either CR or LF inside line values, uses CONT records to communicate a line feed
- respects the maximum line length of 255 code units, uses CONC record to support long line values
- *does not* use CONC or CONT records in the GEDCOM header
- *never uses* CONC records for line values with a maximum length of 248 or less
- writes the submitter record directly after the GEDCOM header
- uses legal GEDCOM 5.5.5 records exclusively
- does not create any empty lines or leading white space
- Makes sure that every line value is a valid string in the encoding used, by not merely splitting lines between code units or code, but between characters

GEDCOM 5.5.5 Reader

- *must* demand that GEDCOM file starts with a Byte Order Mark (BOM), *must reject* the file as not GEDCOM 5.5.5, with a message such as “ostensible GEDCOM 5.5.5 file lacks Byte Order Mark”
- *must* demand a legal character encoding, *must not* tolerate illegal character sets or encodings
- *must* accept UTF-8, but *should* detect and reject CESU-8
- *must* check that the HEAD.CHAR value matches the encoding used, and abort with a fatal error when it does not
- *must* check that each line value is a valid string in the encoding use, and *must* reject the GEDCOM file if any string is invalid
- *must* reject a GEDCOM header without a GEDCOM form
- *must* check that the HEAD.SUBM pointer, if present, is valid
- *must not* try to process an GEDCOM form it does not explicitly support, but *must* abort with an “unsupported GEDCOM form” message
- *must not* accept lines longer than 255 code units, *must* reject the file as invalid, with an error message such as “Line is too long. This is not a GEDCOM 5.5.5 file.”.
- Accepts all line values *as-is*, *does not* trim leading or trailing white space from any line value
- processes CONC and CONT records correctly
- *must not* tolerate CONC or CONC records in the Basic GEDCOM header
- *must* accept CONC or CONC records in the GEDCOM form's header extension
- *must* demands that system identifiers are valid, *must not* tolerate known nonsense

values

- *must* demand legal records, *must not* tolerate illegal records
- *must* especially demand a valid GEDCOM header, *must not* accept any error in the header
- *must* import data as-is, *must not* modify anything during import
- *must not* tolerate empty lines or leading white space, but *must* reject such files as not GEDCOM.

Support for Earlier GEDCOM Versions

GEDCOM 5.5.5 succeeds GEDCOM 5.5.1 (1999 CE). As the GEDCOM 5.5.1 specification has been the *de facto* standard for two decades, there is no pressing need to support any older GEDCOM versions, not even GEDCOM 5.5.
Applications should take care to recognise GEDCOM 5.5.1 files with a truncated version number; GEDCOM 5.5.1 files that are incorrectly labelled as GEDCOM 5.5 files.

Application that already support GEDCOM 5.5.1, should add support for GEDCOM 5.5.5, while continuing to support GEDCOM 5.5.1. New applications should support GEDCOM 5.5.5 first, and developers should only add GEDCOM 5.5.1 support if there is still significant demand for it. GEDCOM export should default to the latest GEDCOM version.

Applications should maintain existing GEDCOM 5.5.1 export functionality. Applications should not offer export to GEDCOM versions older than GEDCOM 5.5.1.

Applications *must* detect the GEDCOM version of GEDCOM files before trying to read them, and *must not* try to read GEDCOM versions they do not support, but *must* honestly report unsupported versions as unsupported.

Applications should read older GEDCOM versions for compatibility's sake. Every application should be able to read the GEDCOM files it has written, including all GEDCOM files produced by earlier versions of that application, just as it should be able to read old versions of its own file format.

Developers who decide to drop import of GEDCOM versions before GEDCOM 5.5.1, should provide an alternative means to get these imported, for example by providing a free download of an older version that can import the older files, and then export to GEDCOM 5.5.1.

Introduction

GEDCOM is an acronym for **GE**nealogical **D**ata **Com**munication.
That name is a misnomer. GEDCOM neither is nor contains a communication protocol.
GEDCOM is a data file format.

GEDCOM was created by FamilySearch, a subsidiary of the The Church of Jesus Christ of Latter-day Saints (LDS), that was originally known as the Family History Department.

GEDCOM aims to provide a flexible, uniform format for exchanging computerised genealogical data.
Its purpose is to foster the sharing of genealogical information and the development of a wide range of interoperable software products to assist genealogists, historians, and other researchers.

Purpose and Content of *The GEDCOM Standard*

The GEDCOM Standard is a technical document written for computer programmers, system developers, and technically sophisticated users. It covers the following topics:

- Basic GEDCOM Language (see Chapter 1, page 46)
- Lineage-Linked Form (see Chapter 2, page 53)
- Character Sets in GEDCOM (see Chapter 3, page 112)
- Lineage-Linked GEDCOM Tags (see Appendix A, page 121 and Chapter 2, page 53)
- ANSEL Character Codes and ANSEL / Unicode conversion (see Appendix B, page 130 and Appendix C, page 132)

Two Levels

GEDCOM is a two-level specification. The base level, the *GEDCOM data format* is not unlike XML; it defines the basic format and syntax. The second level, a *GEDCOM form*, is not unlike an XML Document Type Definition (DTD); it defines a particular schema on top of the basic rules. Just like in XML, the base level is a bit more than just format and syntax, but also includes items that must be present c.q. supported in every file; this is the Basic GEDCOM Language.

Chapter 1 describes the base level, the ***GEDCOM data format***, both the basic syntax and the Basic GEDCOM Language. This is a general-purpose data representation language for representing any kind of structured information in a sequential medium. Chapter 1a discusses the GEDCOM syntax and identification of structured information in general. It does not deal with the semantic content of any particular kind of data.
Chapter 1b discusses the Basic GEDCOM Language, such as the mandatory header and trailer.
Chapter 2 describes the higher level by way of one particular ***GEDCOM form***, known as the *Lineage-Linked Form*. This is the GEDCOM use by genealogical software systems to exchange data with each other.

The separation between the GEDCOM data format and GEDCOM forms is half-hearted.
The GEDCOM specification does not properly separate the GEDCOM data format and the lineage-linked from another. The GEDCOM header (in which you specify the GEDCOM form) is actually defined as part of the lineage-linked form instead of the GEDCOM data format, where it logically belongs. There is also just one GEDCOM version number, while there really should be two, one for the GEDCOM data format, and one for the lineage-linked form.

Chapter 1 Basic GEDCOM Language

Introduction

This chapter describes the core GEDCOM data representation language.
The generic data representation language defined in this chapter may be used to represent any form of structured information, not just genealogical data, using a sequential stream of characters.

Concepts

A GEDCOM file is a text file that represents a database in the form of a sequential stream of related records. A record is represented as a sequence of tagged, variable-length lines, arranged in a hierarchy. A GEDCOM line always contains a hierarchical level number, a tag, and an optional value. A line may also contain a cross-reference identifier or a pointer.
Each line, including the last one, is terminated by either a carriage return (CR), a line feed character (LF), or the carriage return/line feed combination (CR/LF). GEDCOM allows different line terminators, but each GEDCOM file must use a single line terminator throughout the file. GEDCOM 5.5.1 and earlier allow the line feed/carriage return (LF/CR) combination as a terminator, GEDCOM 5.5.5 does not.

Record types are identified by tags. Each tag identifies a GEDCOM record type. Different subrecord types may use the same tag.
Top-level records are identified by their tag. Subrecords are identified by their tag, but only fully identified by their complete tag hierarchy; all the tags of the enclosing records and the tag of the subrecord itself.

Recording records as tags with values allows record types to occur one, multiple times or zero times. It also allows the introduction of new records and subrecords without introducing incompatibility, as a reading system will continue to process the records it does understand, while ignore records it does not understand. This allows so-called user extensions.

GEDCOM files consists of records that may have subrecords. Subrecords may have subrecords again. A record without subrecords takes up exactly one GEDCOM line. A record with subrecords takes up multiple GEDCOM lines.

Each line is a GEDCOM record. Most one-line records are subrecords of an enclosing record. The enclosing records of a subrecord are known as the context of that record. The meaning of a subrecord depends on it context. A record and the subrecord it encloses are simply known as a record.

The hierarchy of records and subrecords is indicated by the level numbers. Top-level record have level number 0 (zero). Subrecords have a higher level numbers. Immediate subrecords have a level number exactly one higher than the enclosing record. It is a fatal error to skip a level.

A GEDCOM file consists of a sequence of top-level records. Each top-level record takes up one or more lines. Most top-level records have subrecords, so most top-level records consist of multiple lines. The beginning of one top-level record marks end of the previous top-level record. each top-level record continues to the next record with level number 0 or the *end of file* (EOF).

In addition to hierarchical relationships, GEDCOM defines the inter-record relationships that allow a record to be logically related to other records, without introducing redundancy. These relationships are represented by two additional, but optional, parts of a line: a cross-reference pointer and a cross-reference identifier. The cross-reference pointer "points at" a related record, which is identified by a required, matching unique cross-reference identifier. The cross-reference identifier is analogous to a primary key in relational database terminology.

Grammar

This chapter defines the grammar for the GEDCOM format. The grammar is a set of rules that specify the character sequences that are valid for creating the GEDCOM line. The character sequences are described in terms of various combinations of elements (variables and/or

constants). Elements may be described in terms of a set of other elements, some of which are selected from a set of alternative elements. Each element in the definition is separated by a plus sign (+) signifying that the items on either side of the plus sign are to be concatenated. When there is a choice of different elements that can be used, the set of alternatives are listed between opening and closing square brackets ([]), with each choice separated by a vertical bar([alternative_1 | alternative_2]). When there is only one alternative shown then the choice is optional, that is, it is the same as [alternative_1 | <NULL>]. The user can read the grammar components of the selected element by substituting any sub-elements until all sub-elements have been resolved.

A GEDCOM file consists of a sequence of top-level records, which may contain subrecords. Each record consists of a sequence of **gedcom_lines**, all contained in a sequential file or stream of characters. The simplest GEDCOM records consist of just one **gedcom_line**. The following rules pertain to the **gedcom_line**:

Grammar Rules

- Long values can be broken into shorter GEDCOM lines by using a subordinate CONC or CONT tag. The CONC tag assumes that the accompanying subordinate value is concatenated to the previous line value without adding a newline. If a concatenated line value is broken at a space, then the space must be carried over to the CONC line value. The CONT record assumes that the subordinate line value is concatenated to the previous line, after inserting a newline.

  A GEDCOM writer should consistently use the same newline throughout the GEDCOM file, and that newline should be appropriate for the receiving system (`HEAD.DEST`).
  A GEDCOM reader should insert the newline appropriate for the platform it is running on.

- The beginning of a new top-level record is designated by a line whose **level** number is 0 (zero).

- Level numbers must be between 0 to 99 and must not contain leading zeroes, for example, level one must be 1, not 01.

- Each new level number must be no higher than the previous line plus 1.

- All GEDCOM records must have a value, i.e. each GEDCOM record must have a line value, subrecords or both. So, every GEDCOM record without subrecords must have a line value. The two GEDCOM records that are the exceptions to this rule are the CONT and TRLR record; empty CONT records are needed to record a sequence of empty lines, and the mandatory TRLR does not have a line value or subrecords.
  By the way, the CONC record *isn't* an exception to this rule; each CONC record must have a line value.

- Any length constraints are given in code units, not bytes or characters. When wide characters are used, byte buffer sizes should be adjusted accordingly.

  The length of a text is measured in characters, code points or code units, the size of a buffer is measured in bytes.

  The *characters, code units, and code points* annotation provides a brief explanation.

- The cross-reference ID has a maximum length of 22 code units, including the enclosing ‘at ’ signs (@), and it must be unique within the GEDCOM file.

GEDCOM Identifier Length

Here, the GEDCOM grammar states that GEDCOM identifiers are limited to 22 code units (characters), including the enclosing at signs. Further on, the lineage-

linked form states that identifiers are at most 22 characters, but that's 22 *without* the enclosing at signs. Thus, the GEDCOM grammar and the lineage-linked form contradict each other. The GEDCOM grammar takes precedence over any GEDCOM form.

- The maximum length of a GEDCOM identifier is 22 code units.
- The maximum length of the identifying part within the at signs is 20 code units.

References

GEDCOM Identifiers: Length

! A GEDCOM file must have *referential integrity*; The existence of a pointer does not merely imply the existence of a corresponding record, it *demands* the existence of corresponding record. A GEDCOM reader *must* make sure all pointers are valid, and must reject any GEDCOM file containing invalid pointers as corrupt.
Pointers must be to existing cross-reference identifiers for records in the same GEDCOM file, and those records must be of the right record type. Pointers to non-existent cross-reference identifiers are a fatal error. Pointers to existing cross-reference identifiers, but of the wrong record type, are fatal errors too.
Orphaned top-level records, i.e. top-level record that aren't pointed to, are perfectly legal. A GEDCOM reader must import orphaned records just like other records. A smart GEDCOM reader may issue a warning about orphaned top-level records, but should never do so for the special case of a GEDCOM file that contains just one record.

! The length of a GEDCOM tag is a maximum of 31 code units.
All characters of the tag are significant.

! The total length of a GEDCOM line, including level number, cross-reference number, tag, value, delimiters, and terminator, *must not* exceed 255 code units.
An ostensible GEDCOM 5.5.5 file containing longer lines isn't a GEDCOM 5.5.5 file. A GEDCOM 5.5.5 reader *must not* accept longer lines, but *must* reject the file as invalid, with an error message such as “Line is too long. This is not a GEDCOM 5.5.5 file.”.

! Empty lines and leading white space are illegal. Systems generating GEDCOM 5.5.5 files must not create empty lines or start any lines with leading white space. Systems reading GEDCOM 5.5.5 files must not accept empty lines or lines starting leading white space.
A GEDCOM 5.5.5 writer *does not* create any empty lines or leading white; if it does, it isn't a GEDCOM 5.5.5 writer.
A GEDCOM 5.5.5 reader *must not* tolerate empty lines or leading white space, but *must* reject such files as not GEDCOM.

Grammar Syntax

A **gedcom_line** has the following syntax:

gedcom_line:=

level + [delim + xref_ID] + delim + tag + [delim + line_value] + terminator

for example:
1 NAME Will /Rogers/

The components used in the pattern above are defined below in alphabetical order. Some of the components are defined in terms of other primitive patterns. The spaces used in the patterns below are only to set them apart and are not a part of the resulting pattern. Character constants are specified by their Unicode code point, using the notation introduced with Unicode 2.0; a Unicode code point is represented as U+*nnnn*, where *nnnn* is the hexadecimal value, e.g. U+0020 is the Space character. Character constants that are separated by a (-) dash represent any character within that range from the first constant shown to and including the second constant shown.

alpha:=

[U+0041 - U+005A | U+0061 - U+007A]

where:

U+0041 - U+005A = A through Z
U+0061 - U+007A= a through z

alphanum:=

[alpha | digit]

delim:=

space

where:

space = U+0020, the Space character

The explicit inclusion of the delim definition as the space character is no mistake. It stresses that the delimiter is a space character and nothing else.

digit:=

U+0030 - U+0039
where:
U+0030 - U+0039 = One of the digits 0, 1,2,3,4,5,6,7,8,9

escape:=

U+0040 + U+0023 + escape_text + U+0040

where:

U+0040 = @
U+0023 = #

A GEDCOM escape sequence begins and ends with an at sign.
An escape sequence must not contain any at sign and must not be followed by an at sign; an escape sequence must be followed by either a delim (the space character) or a terminator.

escape_text:=

[alphanum | escape_text + alphanum | escape_text + space]

where:

space = U+0020, the Space character

The escape_text is the part of an escape between the opening and closing at sign (@).
Escape sequences are defined by a GEDCOM form for that GEDCOM form.

GEDCOM 5.5.1 allows almost any Unicode character inside escape sequences. GEDCOM 5.5.5 restricts escape sequences to alphanumerical characters and the space character.
Notice that escape_text must start with an alphanumerical character, that it must not start with a space.

The space character is allowed but deprecated; it is only included to keep the already existing escape sequence `@#DFRENCH R@` legal.

identifier_string:=

[alphanum | alphanum + identifier_string]

level:=

[digit | non_zero_digit + digit]

The level number consists of one or two digits. It may be zero, but must not start with leading zeroes.

line_char:=

all legal characters with some exceptions and one special case

specifically disallowed:
U+0000 - U+001F, except U+0009 = most C0 control characters
U+00FF = Delete character

specifically allowed:
U+0009 = Horizontal Tab

special case:
U+0040 + U+0040 = @@

Most control characters in the range U+0000 - U+001F (C0 Control Set) are disallowed. The one exception is U+0009, the Horizontal Tab. Text may contain horizontal tabs.
Note that Carriage Return (U+000D) and Line Feed (U+000A) must not occur inside text. Line breaks are supported through the `CONT` record. The use of the space character ( ), number-sign (#) and underscore (_), is perfectly legal in text.

Legal Characters

All text must abide by the rules of the character set used. The character set used determines which code points are legal. The character set used determines which code point combinations are legal. Use string functions to make sure that any line breaks are between characters, not inside characters.

Special Case: At sign (@)

The At sign (@) is legal in text, but because of its special function in GEDCOM, there's a twist. Text is not allowed to contain a single at sign (@), but must always include two consecutive at signs (@@) instead, to clearly distinguish an at sign that is just text from an at sign starting a cross-reference or escape sequence.
This is similar to backward slashes (\) having to be doubled inside C and C++ text strings.

This at sign rule affects all textual line values, but particularly the EMAIL line value, <ADDRESS_EMAIL>, introduced with GEDCOM 5.5.1, as email addresses always contain a single at sign. Within a GEDCOM file, that single at sign must be represented by a double at sign.

A GEDCOM writer must export a double at sign instead of single at sign, and a GEDCOM reader must import each double at sign as a single one. Exporting a single at sign where a double at sign should be exported is a serious GEDCOM grammar mistake. A GEDCOM 5.5.5 reader *must not* interpret a single at sign as if it were a double at sign, but must report a fatal error and exit.

dates

GEDCOM writers should pay special attention to at signs in date fields. When an application

does not provide support for a particular calendar, a user may choose to enter a date complete with the <DATE_CALENDAR_ESCAPE> for the calendar, e.g. "@#JULIAN@ 12 Oct 1492" (without the quotes). The at signs within such dates should *not* be doubled.

Not Deprecated

This feature complicates GEDCOM handling, so it is desirable to mark this GEDCOM feature as deprecated, and eventually get rid of it, but as long as some records can contain GEDCOM pointers as well as free text, this feature allows GEDCOM readers to distinguish between an at sign that starts a pointer (single at sign), and one that is merely an at sign in some text (double at sign).

line_item:=

[escape | line_text | escape + delim + line_text]

This line_item definition is different from the GEDCOM 5.5.1 definition. It explicitly restricts line_item to just three different formats.

The explicit inclusion of nothing but an escape sequence as a possible line value is as intended. It has been used in GEDCOM 5.0, it is deprecated now.

line_text:=

[line_char | line_text + line_char]

line_text is text that neither is nor contains a pointer or an escape sequence.

line_value:=

[pointer | line_item]

non_zero_digit:=

U+0031 - U+0039
where:
U+0031 - U+0039 = One of the digits 1,2,3,4,5,6,7,8,9

null:=
nothing

pointer:=

xref_ID

tag:=

[[U+005F] + alphanum | tag + alphanum]

where:

U+005F = _ (underscore)

GEDCOM tags are Case-Sensitive.
Each tag has a single correct casing.
All Basic GEDCOM Language tags are UPPERCASE.

- A GEDCOM writer must use the correct casing for tags.
- A GEDCOM reader must recognise tags through case-sensitive comparison.
- A GEDCOM reader must not recognise tags through case-insensitive comparison.

GEDCOM tags are generally restricted to alphanumeric characters. Tags need not start with

a letter, but may start with a digit. Tags may additionally start with an underscore; this is allowed because the Lineage-Linked Form demands that non-standard tags start with an underscore.

terminator:=

[carriage_return | line_feed | carriage_return + line_feed]

where:

carriage_return = U+000D, the Carriage Return character
line_feed = U+000A, the Line Feed character

A GEDCOM writer must not use different terminators for different lines.
A GEDCOM writer must use a single terminator throughout a GEDCOM file.

A GEDCOM file is a text file, which you can load into an text editor. Like any other text file, a GEDCOM file must use a single line terminator throughout the file, and the occurrence of either ᶜᴿ or ᴸᶠ outside the line terminator is a fatal error: not a GEDCOM file, not even a text file.

A GEDCOM writer should not pick the terminator appropriate for the platform it runs on, but the terminator appropriate for the destination system (`HEAD.DEST`). GEDCOM writers should use ᶜᴿ/ᴸᶠ when unsure which terminator to pick. Web applications should always use ᶜᴿ/ᴸᶠ.

Starting with GEDCOM 5.5.5, the terminator must be either ᶜᴿ, ᴸᶠ or ᶜᴿ/ᴸᶠ and may no longer be ᴸᶠ/ᶜᴿ.
This is correction of a mistake; ᴸᶠ/ᶜᴿ should never have been included as an option; its inclusion was a problem instead of a solution. Text editors and GEDCOM readers expect ᶜᴿ, ᴸᶠ and ᶜᴿ/ᴸᶠ, and are likely to get seriously confused when presented with ᴸᶠ/ᶜᴿ. Even though it has been legal for decades, not a single genealogy applications uses ᴸᶠ/ᶜᴿ as a GEDCOM line terminator. The ᴸᶠ/ᶜᴿ terminator only appears in test files specifically crafted to test support for it.

References

- GEDCOM Lines
- Famberry GEDCOM

xref_ID:=

at + identifier_string + at

where:

at = U+0040, the At Sign (@)

GEDCOM 5.5.5 demands that the identifier_string be alphanumerical.

Description of Grammar Components

alpha:=

The alphabetic characters, both uppercase and lowercase, i.e. A through Z and a through z.

delim:=

The **delim** (delimiter), a single space character, terminates both the variable-length **level** number and the variable-length **tag**. Note that space characters may also be present in a

value.

escape:=

The **escape** is a character sequence in the grammar used to specify special processing, such as for switching character sets or for indicating an inclusion of a non-GEDCOM data form into the GEDCOM structure.

An escape sequence should be separated from any line item following it by a delimiter (space character). The delimiter following the escape sequence is not part of the escape sequence (the GEDCOM 5.5.1 grammar erroneously states that it is). Failure to include a delimiter between an escape sequence and a line item following it is a syntax error.

escape_text:=

Legal escape_text values and their meaning are defined by each GEDCOM form for that GEDCOM form.

level:=

The **level** number works the same way as the level of indentation in an indented outline, where indented lines provide detail about the item under which they are indented. A line at any level L is enclosed by and pertains directly to the *nearest preceding line* at level L-1. The level L may increase by 1 at most. Level numbers *must not* contain leading zeroes, for example level one must be `1`, not `01`.

The enclosed subordinate lines at level L are said to be in the context of the enclosing superior line at level L-1. The interpretation of a **tag** must be in the context of the **tag**s of the enclosing line(s) rather than just the tag by itself.
Take the following record about an individual's birth and death dates, for example:

0 INDI
1 BIRT
2 DATE 12 MAY 1920
1 DEAT
2 DATE 1960

In this example, the expression DATE 12 MAY 1920 is interpreted within the INDI (individual) BIRT (birth) context, representing the individual's birth date. The second DATE is in the INDI.DEAT (individual's death) context. The complete meaning of DATE depends on the context.

GEDCOM writers *must not* insert empty lines to highlight the end of one and beginning of another record nor indent lines based on their level number to increase readability. The GEDCOM grammar is crystal clear that GEDCOM files do not contain empty lines and GEDCOM lines do not contain leading white space.
GEDCOM 5.5.5 readers must reject any such files as *not a GEDCOM file*.

Anyone viewing or editing a GEDCOM file can enjoy increased readability though colour coding, indentation and other conveniences by using a dedicated GEDCOM editor or text editor with GEDCOM support.

line_char:=

Characters legal within a textual line value.

line_text:=

A purely textual line value (no pointers or escape sequences).

line_value:=

The **line_value** identifies an object within the domain of possible values allowed in the context of the **tag**. The combination of the tag, the **line_value**, and the hierarchical context of the supporting **gedcom_line**s provides the understanding of the enclosed **value**s. This

domain is defined by a given GEDCOM form. The lineage-linked form defined in Chapter 2 Lineage Linked-Form, page 53 is one such GEDCOM form.

Values whose source information contains illegible parts of the value should be indicated by replacing the illegible part with an ellipsis (...).

Values are generally not encoded in binary or other abbreviation schemes for reducing space requirements, and they are generally constrained to be understandable by a typical user without decoding. This is intended to reduce the decoding burden on the receiving software.

GEDCOM files are text files with a fair amount of repetition, and as such compress well using common file compression methods.

Line Value Ellipsis

The instructions to use ellipsis for illegible parts is problematic, because original text may contain ellipsis already.

FamilySearch probably meant this instruction for names in particular; use ellipsis for illegible parts of a name.

The **line_value** within the context of a tag hierarchy of **gedcom_line**s represents one piece of information and corresponds to one field in traditional database or file terminology.

pointer:=

A **pointer** stands in the place of the record or context identified by the matching **xref_ID**. The use of **pointer**s is explicitly defined within the GEDCOM form, such as the Lineage-Linked GEDCOM Form defined in Chapter 2 (p. 53).

The **pointer** is a link to a top-level records with an identifier inside the same GEDCOM file. The GEDCOM grammar does not support linking to records in other files or to subrecords.

A pointer must be both legal and valid. A pointer is legal if it conforms to the pointer syntax. A pointer is valid if it matches the identifier of a record and that record is of the right type.

The pointer and matching record need not appear in any particular order; both *backward references* (record appeared already) and *forward references* (pointer appears first) are legal. GEDCOM readers typically create a look-up table of record identifiers while reading a GEDCOM file, and verify the validity of pointers against that table after reading the entire file.

The GEDCOM 5.5.1 specification mentioned two possible future features, for which it already complicated the parsing of pointers. The two obsolete future features are linking to records in other files, and linking to subrecords, which demand that GEDCOM readers give special treatment to colons and exclamation marks respectively.
The GEDCOM 5.5.5 specification does not allow semicolons or exclamation marks in pointers.

Verifying whether pointers are legal was complex already, as GEDCOM 5.5.1 allowed many different characters inside pointers. GEDCOM 5.5.5 simplifies pointers by restricting pointers to alphanumeric characters.

tag:=

A **tag** consists of a variable length sequence of **alphanum** characters. All user-defined tags that have not been defined in the GEDCOM standard, must begin with an underscore character (U+005F.

A tag defines the meaning of the **line_value** within the context of the enclosing tags, and

contributes to the meaning of the enclosed subordinate lines. Specific **tag**s are defined in Appendix A, page 121.
The presence of a tag together with a value represents an assertion which the submitter wishes to communicate to a receiver. A tag without a value does not represent an assertion. If an optional record (the entire line; level, *tag*, line value and terminator) is absent, no assertion is made. Information of a negative nature (such as knowing positively an event did not occur) is handled through the semantic definition of a different tag and its accompanying value to assert that information explicitly.

Although all the tags of the lineage-linked form are only three, four or five characters long, systems must be able to handle all legal tags.
The GEDCOM 5.5.1 specification demanded that tags be unique in the first 15 characters (code units), and thus allowed for wild variation beyond the first 15 characters. The GEDCOM 5.5.5 specification simplifies GEDCOM parsing by demanding that all tags are unique, period. For maximum backward compatibility, developers are still advised to avoid tags that aren't unique in the first 15 characters.

How different record types and their tags may be combined is defined by a GEDCOM form. A GEDCOM form defines the valid record and subrecords, their line values and possible cross-references.

Valid **tag**, **line_value**, **xref_ID**, and **pointer** combinations are constrained by the GEDCOM form defined for representing a given kind of information. Chapter 2 Lineage-Linked Form, page 53 defines the commonly used GEDCOM form.

Tags and records aren't the same thing. The FamilySearch GEDCOM specifications sometimes uses “tag” to mean “record”. That is confusing. A conscious effort has been made to always use the right term throughout the GEDCOM 5.5.5 specification.

terminator:=

The **terminator** delimits the variable-length **line_value** and signals the end of the **gedcom_line**.
The valid terminators are Carriage Return (CR), Line Feed (LF) and the CR/LF combination. The LF/CR combination is no longer allowed.

GEDCOM writers must pick one line terminator, and use only that line terminator throughout the entire GEDCOM file.

xref_ID:=

(See **pointer**), page 35
The **xref_ID** is formed by any arbitrary combination of alphanumeric characters. No meaning is attributed by the receiver to any part of the **xref_ID**, other than its **unique** association with the associated record.

Examples:
The following are examples of valid but unrelated GEDCOM lines:

```
0 @I1234@ INDI
. . .
1 AGE 13y
. . .
1 CHIL @I1234@
. . .
1 NOTE This is a note field that is
2 CONT continued on the next line.
```

The first line has a **level** number 0, a **xref_ID** of @I1234@, an INDI **tag**, and no **value**.
The second line has a **level** number 1, no **xref_ID**, an AGE **tag**, and a **value** of 13y (13 years).
The third line has a **level** number 1, no **xref_ID**, a CHIL **tag**, and a **value** of a **pointer** to a **xref_ID** named @I1234@.

History: FamilySearch GEDCOM 5.5.1 allowed many different characters inside pointers, even spaces, and demanded special handling of colons and exclamation marks in support of obsolete future features.
GEDCOM 5.5.5 simplifies parsing by limiting pointers to alphanumeric characters.

Implementation note: In practice all systems use table indices to generate unique identifiers; the identifier for the first record in the individual table is @I1@, the identifier for the fourth record in the sources table is @S4@, and so on.
The receiving system may ignore this numbering system, but often chooses to use the same indices as used in the source system. Users of multiple applications like this, as it allows searching for individuals and family groups using the same numbers as used in the source system.

References

- Common GEDCOM Identifier Naming Convention

White Space

Spaces & Tabs

For purposes of the GEDCOM grammar, *white space* is defined as spaces and tabs (horizontal tabs, vertical tabs are illegal). White Space includes spaces and tabs only, line terminators are not considered white space.

Unicode has many more characters in the white space category, but those characters are not relevant to GEDCOM parsing. GEDCOM readers and writers *must not* take any special action for those characters.

GEDCOM 5.5.1 versus 5.5.5

GEDCOM 5.5.1 did not allow tabs, but many GEDCOM 5.5.1 files do contain tabs anyway. Few genealogy software developers ever restricted user from entering horizontal tabs, mostly because they did not even notice the restriction.
GEDCOM 5.5.5 allows user text to contain horizontal tabs.

Previous GEDCOM versions do not explicitly allow or disallow trailing white space on GEDCOM lines. GEDCOM 5.5.1 mentions that many systems will strip trailing spaces when reading GEDCOM lines, and mentions it as fact that should be taken into account, which seems a tacit approval of that practice.
GEDCOM 5.5.5 changes that: a GEDCOM reader *must not* strip trailing white space.

Superfluous versus Significant

This specification distinguishes between superfluous and significant white space.
Superfluous white space is white space that serves no purpose, and really should not be there, such as white space before or after a place name in a place name field. Superfluous white space is often the result of an edit error, and its continued presence is a normalisation issue. Superfluous white space can and should be trimmed from application records.
Significant white space is white space that must not be lost, such as a tab or space between two words. Significant white space includes all white space entered into free-form text fields, regardless of senseless some of that white space may seem to a human.

GEDCOM Lines

GEDCOM lines *must not* start with leading space.

GEDCOM lines *should not*, but *may* contain trailing white space.

GEDCOM lines *should not* have trailing white space because most line values should not have trailing white space.
Things like names, dates and places should have neither leading nor trailing white space, and it is the job of the GEDCOM writer, or rather the application it is part of, to prevent such leading or trailing white space from occurring.

GEDCOM lines *may* contain trailing white space because it cannot always be avoided. The pathological case of a text consisting of lots of white space characters and few newlines forces a GEDCOM writer to create CONC & CONT records that end with white space. A GEDCOM reader must accept those line values as-is, not remove any white space, to correctly reconstitute the original text.

Trailing White Space

GEDCOM lines *must not* not end with superfluous trailing white space.
GEDCOM lines *may* contain significant trailing white space.

A GEDCOM writer, or rather the system it is part of, must ensure that it does not write superfluous white space, but only significant white space. For many fields, applications should automatically strip any leading and trailing white space upon user input. Developers should include warnings about leading and trailing white space as part of the consistency checks.

A GEDCOM reader must import the lines values presented by a GEDCOM file as they are. A GEDCOM reader must treat every part of a line value, including all white space, as significant. **A GEDCOM reader *must* import every line values as-is, *must not* not strip trailing white space.**

Leading White Space

The rules for leading and trailing white space are not really rules about GEDCOM lines, but about GEDCOM lines values. Line values *should not*, but *may* contain trailing white space. The same goes for leading white space: Line values *should not*, but *may* contain leading white space.

A line value may start with leading white space. That CONC line values starts with leading white space is often deliberate. A GEDCOM reader must import the lines values presented by a GEDCOM file as they are. A GEDCOM reader must treat every part of a line value, including all white space, as significant. **A GEDCOM reader *must* import every line values as-is, *must not* not strip leading white space.**

CONC & CONT

GEDCOM Grammar

The CONC & CONT records are defined as part of the GEDCOM grammar. They are always available for use with any line value of any record in any GEDCOM form.

The CONC & CONT records must not be used with records defined in the GEDCOM grammar. They may only be used with records defined in a GEDCOM form.
There is no such thing as CONC or CONT records subordinate to CONC or CONT records.

The CONC & CONT record always appear as immediate subrecords of a GEDCOM form record. There is no such thing as a top-level CONC or CONT record.

GEDCOM 5.5.5 Rule

The GEDCOM 5.5.5 specification introduces a new rule

While it is allowed to use CONC records for any line value, GEDCOM writers *should* not use them needlessly. GEDCOM writers *should not* use CONC records for line values with a maximum length of 248 code unit or less.

GEDCOM 5.5.5 introduces one exception to the rule that CONC & CONT records may be used with any records; the CONC & CONT records *must not* be used in the GEDCOM header.

Logical versus Physical Line Value

This text distinguishes between logical line values and physical line values. The GEDCOM grammar defines physical line values, and a GEDCOM form defines logical values:

- The GEDCOM grammar defines physical line values. A physical line value may be at most 248 code units long, and *must not* contain any carriage return (CR) or line feed (LF) character.
- A GEDCOM form defines logical values. A single logical line value may be a long text consisting of multiple lines. A single logical line value may be up to 32.765 code units long, and may contain newlines.

The CONC & CONT records exist to bridge the differences. The CONC & CONT records enable support of logical line values despite the limitations of the physical lines values.

Line Terminator versus Newline

This text refers to the line break at end of a GEDCOM line as a *line terminator*, and a line break inside a logical line value as a *newline*. This convention should make the text easier to understand, and allows making the following point, which is hard to make otherwise: the line terminator used for GEDCOM lines and the newline used within logical line values need not be the same. For example, a web application running on a Unix server would typically use line feed (LF) as its newline inside the logical line value, and the carriage return/line feed (CR/LF) combination as the line terminator for its GEDCOM lines.

CONC

The CONC (concatenation) record allows a single logical line value to split over multiple GEDCOM lines. This is needed because a GEDCOM line must not be more than 255 character long, and GEDCOM must still be able to handle logical line values longer than that.
It is called the concatenation record because a GEDCOM reader must concatenate the physical line value of a CONC record to the logical line value of the superior record to reconstitute the original logical line value.

CONT

The CONT (continuation) record is how GEDCOM supports multiple lines *within* a logical line value. This is needed because a logical line value may contain newlines, but a physical line value must not contain any carriage return (CR) or line feed (LF) characters. It is called the continuation record because a GEDCOM reader must continue on a new line (by reinserting the newline that prompted the GEDCOM writer to use a CONT record) before appending the physical line value of the CONC record to the physical line value of the superior record.

Usage

Usage of the CONC & CONT records is simple. A GEDCOM writer splits a logical line value into parts as needed, and writes those parts as the physical line values of a subordinate CONC or CONT record. A GEDCOM reader reconstitutes the logical line value by recombining the line value of the record with the line values of the CONC & CONT subrecords.

CONC Usage

A GEDCOM writer creates another CONC record whenever the remaining logical line value is too long for the current physical record.
A GEDCOM reader simply appends the physical line value of the CONT record to the logical line record is it reconstituting, *without adding anything in between*.

CONT Usage

A GEDCOM writer creates a new CONT record whenever it encounters a newline, then continues writing the rest of the logical line value.
When a GEDCOM reader encounter a CONT record, it appends a newline to the logical line value it is reconstituting before appending the physical line value of the CONT record.

Character and Code Point Boundaries

A GEDCOM writer cannot split a line wherever it likes, and certainly not at some fixed number of code units. After every split, the two resulting parts must both be valid string for the encoding used.

Text-splitting must respect *code point* boundaries. Many Unicode code points require multiple code units, not only when UTF-8 is used, but also when UTF-16 is used. Text-splitting *must not* split at just any code unit, as that might split the text *inside* a code point. Text-splitting must occur *between* code points.

Text-splitting must respect *character* boundaries. Unicode support characters that consists of multiple code points. Text-splitting *must not* split at just any code point, as that might split the text *inside* a character. Text-splitting must occur *between* characters.

Developers *must not* simply split a line value at a fixed number of code units, or some fixed number of code points, they must ensure that they split between characters. Developers need not implement their own string manipulation routines to get this right. Developers can and should avoid string handling problems by taking advantage of the string routines offered by the their programming language's library, framework, platform or operating system. These routines know how to split Unicode strings.

The ANSEL character set allows characters consisting of multiple code points too. There are no library routines that perform correct splitting of ANSEL strings, but that does not matter, as there is no need to split ANSEL strings anymore. The *GEDCOM 5.5.1 Annotated Edition* recommends using Unicode, the GEDCOM 5.5.5 specification demands Unicode.
Developers do not need to write ANSEL at all, and only to read ANSEL if they want to read GEDCOM 5.5.1 and earlier files, including ANSEL GEDCOM files.

Practical Problems

Conceptually, using CONC & CONT is simple. Historically, there are two practical problems:

- The GEDCOM writer must figure out where to break the logical line value.
- The GEDCOM reader must decide what to do with leading and trailing white space.

Trailing White Space

The GEDCOM specification does not forbid trailing white space, but for most records, trailing white space is meaningless. Historically, many GEDOM readers strip trailing white space from records.
That is problematic, as the trailing white space on a CONC or CONT record may be the space between words that would be erroneously concatenated if that space were lost.

That is why GEDCOM 5.5.1 requires that a split be done either in the middle of word, or just

before a space. When the split occurs just before a space, that space becomes the first character of the line value on the next line. Such leading white space is considerably less likely to be lost than trailing white space. It might still get lost because some GEDCOM readers skip all white space after a tag, instead of skipping just one space as they should.

The CONC or CONT annotation in the *GEDCOM 5.5.1 Annotated Edition* discusses the problems with and solutions for GEDCOM 5.5.1 and earlier in some detail.

GEDCOM 5.5.5 specifies the simplest possible solution to all the CONC or CONT complications. It is straightforward rule that GEDCOM 5.5.5 readers must follow anyway: a GEDCOM reader must import line values correctly.
A GEDCOM reader must import the lines values presented by a GEDCOM file as they are. A GEDCOM reader must treat every part of a line value, including all white space, as significant. A GEDCOM reader *must not* trim leading or trailing white space from line values. A GEDCOM reader must simply import the line values as-is, including all leading and trailing white space.

GEDCOM 5.5.5 Writer Best Practice

- Use operating system or programming library functions to make sure you split between characters, not inside characters
- Otherwise, split wherever is convenient; this GEDCOM file will be read by a GEDCOM 5.5.5 or later reader

GEDCOM 5.5.5 Reader Rules

- import each line value *as-is*
- *do not* trim trailing white space from any GEDCOM line or line value
- *do not* trim leading white space from any line value

References

- GEDCOM CONC and CONT
- Gaenovium Presentations: Louis Kessler: Reading Wrong GEDCOM Right
- Behold blog 2010-01-10: CONC Me on the Head

Symbols Used with Basic GEDCOM Language and GEDCOM forms

The following symbols are used definitions of the Basic GEDCOM Language and GEDCOM forms:

<<double_angle bracket>>
The name within the brackets is a subordinate structure of one or more GEDCOM records (and subrecords) that is to be substituted in place of the line containing the enclosing double angle brackets.

<Single_angle bracket>
The name within the brackets is a basic value type, also known as a primitive.

{braces}
Indicates the minimum and maximum occurrences allowed for this structure or line—{Minimum:Maximum}. Note that minimum and maximum occurrence limits are defined relative to the enclosing superior line. This means that a required line (minimum = 1) *is not required if the optional enclosing superior line is not present*. Similarly, a line occurring only once (maximum = 1) may occur multiple times as long as each occurs only once under its own multiple-occurring superior line.

For textual line values, the numbers specify the minimum and maximum length of the *logical line value*. Long logical line values may require the use of CONC & CONT records

to keep all physical line values below the maximum of 255 code units.

When the minimum and maximum are identical, a shorthand is used; the definition simply gives the size, instead of repeating it with a colon in between, so {Size=3} instead of {Size=3:3}.

[Square brackets]
Indicates a choice of one or more options— **[**Choice of**]**.

vertical | bar
Separates the multiple choices, for example [Choice 1 | Choice 2].

n level number
A level number which assumes the level number of the line which referenced the substructure name.

+1, +2 ...
A **+1** level number is 1 greater than the level number assumed by the superior **n** level. A **+2** level number is 2 greater, and so forth.

0xHH
Indicates an allowable hexadecimal character value where HH is that value, for example, 0x20 (decimal 32) indicates the space character.

Chapter 1 Part II: Basic GEDCOM Language

The Basic GEDCOM Language is the GEDCOM language shared by all GEDCOM forms. It defines essential GEDCOM records and the overall structure of a GEDCOM file.

Basic GEDCOM

Four Record Types

The GEDCOM Language defines four record types (the header record has a few subrecords):

- **HEAD**: a *header* record to provide basic information about a GEDCOM file.
- **TRLR**: a *trailer* record to signal the end of a GEDCOM file.
- **CONC**: a *concatenation* record to support long line values.
- **CONT**: a *continuation* record to support newlines.

The header record provides just the essentials: GEDCOM version, character encoding, GEDCOM form and form version. The GEDCOM form may extend the header with additional information of relevance to the form.

The trailer record is technically superfluous, but its presence reassuring; its absence allows reporting of incomplete GEDCOM files. It was necessary for when GEDCOM still supported multi-volume GEDCOM files.

The CONC & CONT address limitations of the GEDCOM grammar. They are defined as part of the GEDCOM Language for used with record defined in GEDCOM forms.

GEDCOM File High-Level Structure

A GEDCOM file starts with a header, ends with trailer record, and has zero or more GEDCOM form-specific record in between. A specific GEDCOM may demand a minimum number of form records, the GEDCOM language does not.

GEDCOM_FILE:=

This is the high-level structure of a GEDCOM file: a GEDCOM file starts with a GEDCOM header, which may be extended by the GEDCOM form. The header is followed by any number of form-specific records, and the file is terminated with a trailer record.
The header and trailer record are mandatory, the header extension and form records are *conceptually* optional.

0 <<GEDCOM_HEADER>>	{1:1}	p. 50
+1 <<GEDCOM_FORM_HEADER_EXTENSION>>	{1:1}	p. 49
0 <<FORM_RECORDS>>	{1:1}	p. 48
0 <<GEDCOM_TRAILER>>	{1:1}	p. 51

The above definition demands the presence of <<GEDCOM_FORM_HEADER_EXTENSION>>, to leave the actual demand up to each GEDCOM form. If the <<GEDCOM_FORM_HEADER_EXTENSION>> were optional in this definition, no GEDCOM form could demand the presence of its extensions. A GEDCOM form can go without any GEDCOM header extensions by defining its <<GEDCOM_FORM_HEADER_EXTENSION>> to be empty.

Similarly, the above definition demands the presence of <<FORM_RECORDS>>, to leave the actual demand up to each GEDCOM form.

GEDCOM Record Definitions

CHARACTER_ENCODING:= **{Size=5|7}**

[UTF-8 | UNICODE | ANSEL | ASCII]

A coded value (exhaustive enumeration) that represents the character set and encoding to be used to interpret this data.
The name of the UNICODE value is confusing, it should have been called UTF-16.

The ASCII and ANSEL character sets are obsolete; the ASCII and ANSEL encoding are *not* legal in GEDCOM 5.5.5.
Unicode-based systems creating GEDCOM 5.5.5 files *must not* offer users the ability to export using non-Unicode encodings, as such exports are practically sure to lose information.
It is strongly recommended that applications support *import* of ASCII and ANSEL GEDCOM files for GEDCOM 5.5.1 and earlier.

All GEDCOM 5.5.5 files *must* use Unicode, using either the UTF-8 or UTF-16 encoding.
All GEDCOM 5.5.5 files, UTF-8 as well as UTF-16 GEDCOM files, *must* start with a Byte Order Mark (BOM).

Western applications should default to using the UTF-8 encoding, Eastern application should default to using UTF-16 encoding.
Application *need not* offers user a choice between these two options.

A GEDCOM 5.5.5 reader *must not* tolerate anything but the UTF-8 and UTF-16 encodings, but *must* reject files using any other encoding as not-GEDCOM, specifically as not even having a valid GEDCOM header.
A GEDCOM 5.5.5 reader *must not* accept a GEDCOM file without a BOM, but *must reject* such files file as *not* GEDCOM 5.5.5, with a message such as “ostensible GEDCOM 5.5.5 file lacks Byte Order Mark”.

A developer may choose to always use just one legal Unicode encoding on export of GEDCOM 5.5.5, but *must* support import of all legal Unicode encodings. A GEDCOM 5.5.5 reader *must* support both UTF-8 and UTF-16 GEDCOM files. The GEDCOM import must also support both Little-Endian and Big-Endian UTF-16.

Unicode applications generally UTF-16 internally, so adding UTF-16 support is fairly straightforward. Support for both Little-Endian and Big-Endian UTF-16 takes no more than a byte swap for every word.
However, even UTF-16 text *must not* be imported with a simple copy operation. Different operating systems use different Unicode normal forms. For example, Windows uses Normal Form C, while Mac OS X uses Normal Form D. Applications should always use library, framework or operating system string routines, to ensure their Unicode text has the proper normal form.

Code-page applications must not use GEDCOM 5.5.5, but must continue to use GEDCOM 5.5.1, and should default their GEDCOM export to UTF-8 instead of ANSEL.

GEDCOM 5.5.5 Character Encoding Rules

GEDCOM 5.5.5 Writer

- Use UTF-8 and UTF-16 only
- Western applications should default to UTF-8
- Eastern applications should default to UTF-16

GEDCOM 5.5.5 Reader

- demand that file starts with a Byte Order Mark (BOM)
- demand that the encoding is either UTF-8 or UTF-16
- must support both UTF-8 and UTF-16 GEDCOM files
- must support both Little-Endian and Big-Endian UTF-16 GEDCOM files
- reject files using anything else as not-GEDCOM, not even a valid GEDCOM header

GEDCOM 5.5.1 Detection

The introduction of GEDCOM 5.5.5 does not change the detection of GEDCOM 5.5.1 files. GEDCOM 5.5.5 files must be identified as GEDCOM 5.5.5 files; if they are not identified as GEDCOM 5.5.5, they aren't GEDCOM 5.5.5 files.
Any file that claims to be a GEDCOM 5.5 file is either a GEDCOM 5.5.1 file (most likely) or an actual GEDCOM 5.5 file. It is never a GEDCOM 5.5.5 file.

Any GEDCOM file that claims to be a GEDCOM 5.5 file but uses the UTF-8 encoding must be recognised as a GEDCOM 5.5.1 file with a truncated version number.

References

- GEDCOM Character Encodings
- GEDCOM Version detection

FORM_RECORDS:= **{1:1}**

GEDCOM-form specific records. These top-level records are defined by the GEDCOM form specified in GEDCOM_FORM.

Notice that the <<GEDCOM_FILE>> definition does not contain a <<FORM_RECORD>> (singular) that is allowed to occur many times, but a <<FORM_RECORDS>> (plural) that is allowed to occur just one.
This allows a GEDCOM form to not only define GEDCOM records types, but their order as well.

GEDCOM_FORM:= **{Size=1:20}**

A value that identifies the GEDCOM form used in this GEDCOM file. The value must be alphanumerical string. This string is case-sensitive.

The GEDCOM grammar does not define any GEDCOM form. The identifier for a GEDCOM form is defined by the specification for that GEDCOM form.
Chapter 2 of this specification defines the Lineage-Linked Form, identified by the value `LINEAGE-LINKED`.

Validity Check

A GEDCOM 5.5.5 reader *must* check the specified GEDCOM_FORM. It does so by performing a case-sensitive check against the list of known GEDCOM forms it supports. A GEDCOM reader *must not* attempt to correct for casing or spelling errors in the GEDCOM form.

A GEDCOM reader must reject any file without a <GEDCOM_FORM> as not a GEDCOM file. *A GEDCOM reader may only try to process a GEDCOM file it if recognises and supports the GEDCOM form used.* If a GEDCOM reader does not recognise the GEDCOM form, it *must* abort with a message such as “Unrecognised GEDCOM form”. If a GEDCOM reader recognises the GEDCOM form, but does not support it, it *must* abort with message such as “Unsupported GEDCOM form”.

GEDCOM_FORM_HEADER_EXTENSION:= **{1:1}**

Zero or more additional HEAD subrecords, as defined by the GEDCOM form specified in HEAD.GEDC.FORM.

GEDCOM Records

GEDCOM_VERSION_NUMBER:= **{Size=3:11}**

MMM + dot + mmm [+ dot + rrr]

where:

MMM = 1 through 3 digits; the major version number
mmm = 1 through 3 digits; the minor version number
rrr = 1 through 3 digits; the revision number
dot = (2E), the full stop character

The version number of the specification used.

Version Number is a Contract

There are two GEDCOM version numbers; one for the GEDCOM specification, and one for the GEDCOM form used.
Both versions numbers are contracts; a promise that everything in the GEDCOM file conforms to that GEDCOM specification and that GEDCOM form, that the GEDCOM file only contains records that are legal according to those specifications.

Three Values

A GEDCOM version consists of at least two and at most three dot-separated values in *major.minor.revision* format. The three parts of the *major.minor.revision* format are:

major = the major version number; starts at 0 (zero).
minor = the minor version number; starts at 0 for each new *major* version.
revision = the revision number; starts at 0 for each new *major.minor* version.

Dot-Separated Values

The full stops used in version numbers are known as dots, but pronounced as “point”. A version number never begins or ends with a dot, it always begins and ends with a digit. Values used must appear in the order shown. The major and minor version number are both mandatory. The minor version number must included, even if it is zero.
The revision number may be left off when it zero. When missing, it must be assumed to be zero.

The *major* version number signals a major new release, often including major new features or breaking changes. The *minor* version number version number signals a significant update of the same release, often including some significant new features, but no breaking changes. The *revision* number signals a lesser update, typically including only minor changes.

Each value consists at most 3 digits. Zeroes are allowed, leading zeroes are not; `0.99` is a valid version number, and so is `1.0`, but `1.01` is not. Trailing zeroes must not be left out and aren't implied; version `2.1` and version `2.10` are two different versions, and version `2.1` comes before version `2.9`. The minimum version number is `0.1`. The maximum version number is `999.999.999`.

Older GEDCOM Versions

It is strongly recommended that systems that continue to export GEDCOM 5.5.1 or earlier

files in addition to GEDCOM 5.5.5 files, comply with the GEDCOM 5.5.5 version number rules for all supported versions.

References

- Truncated GEDCOM Version
- GEDCOM Version Detection

GEDCOM_HEADER:=

U+FEFF (Byte Order Mark)

n **HEAD**		{1:1}	
+1 **GEDC**		{1:1}	
+2 **VERS** <GEDCOM_VERSION_NUMBER>		{1:1}	p. 49
+2 **FORM** <GEDCOM_FORM>		{1:1}	p. 48
+3 **VERS** <GEDCOM_VERSION_NUMBER>		{1:1}	p. 49
+1 **CHAR** <CHARACTER_ENCODING>		{1:1}	p. 47

...

Essential Information

This is the definition of a basic GEDCOM header.
The basic GEDCOM header provides just the essential information; everything that's needed for a system to figure out whether it can read this GEDCOM file, no more, no less.
Every GEDCOM file must start with this header.
The continuation dots indicate that the header may be extended by the GEDCOM form.
The GEDCOM header is always preceded by the Byte Order Mark (BOM), Unicode code point U+FEFF.

Valid Header

It is important that GEDCOM 5.5.5 writers produce correct GEDCOM files, and especially important that each file contains a valid GEDCOM header.
A GEDCOM 5.5.5 reader must demand a flawless GEDCOM header; it *must not* process a GEDCOM file if the header is invalid, but must issue a fatal error and abort.

No Modifications

Developers *must not* modify or extend this record in any way.
No additional records may occur, not even CONC or CONT records.

HEAD.CHAR Record required

The HEAD.CHAR record is required.
GEDCOM 5.5.5 demands that GEDCOM files use Unicode, allows only the UTF-8 and UTF-16 encodings, and demands that each GEDCOM file starts with a Byte Order Mark (BOM). That Byte Order Mark already tells the GEDCOM what the encoding of the GEDCOM file is. So, technically, the HEAD.CHAR record is superfluous now. The HEAD.CHAR record remains mandatory for maximum compatibility with GEDCOM 5.5.1 and earlier.

GEDCOM Form Header Extension

GEDCOM forms may extend the Basic GEDCOM header with additional records to create a form-specific header. The GEDCOM form can only *extend* the GEDCOM header; all the additional records must come after the records of the basic GEDCOM header. Use of CONC

nor CONT records is allowed there, but strongly discommended.

Explicit Support required

A GEDCOM writer *must* provide correct **GEDC.VERS** and **GEDC.FORM** values. Systems which process GEDCOM files *must* check theses values, and may only try to read GEDCOM versions and forms it explicitly supports. For GEDCOM versions or forms it does not support, the system must state so and abort, it *must not* try to import a GEDCOM version or form it does not explicitly support.

Form Version

GEDCOM 5.5.5 introduces the HEAD.GEDC.FORM.VERS, a separate version number for the GEDCOM form, in addition to HEAD.GEDC.VERS, the general GEDCOM version number. HEAD.GEDC.FORM.VERS record is mandatory.
Systems *must not* try to read GEDCOM 5.5.5 files if they do not explicitly support GEDCOM 5.5.5, but if a system that only knows about GEDCOM 5.5.1 and earlier tries to do so anyway, and follows the rule that unrecognised records may be ignored, it will probably ignore the HEAD.GEDC.FORM.VERS record.

GEDCOM_TRAILER:=

0 **TRLR** {1:1}

The GEDCOM trailer marks the end of a GEDCOM file.
The TRLR record has neither a line value nor subrecords.
The TRLR record ends with a line terminator *just like any other record*. That terminator must not be absent, and the TRLR record must not be followed by anything else.

GEDCOM 5.5.1 still needed the TRLR record in support of multi-volume GEDCOM files, GEDCOM 5.5.5 no longer allows multi-volume GEDCOM files.
Technically, the TRLR *is* superfluous now, but it remains for backward compatibility and because it's is reassuring; the TRLR record confirms that the end of a complete GEDCOM file has been reached.

GEDCOM Tag Definitions

CHAR {CHARACTER_ENCODING}:=

The character set and encoding used in this GEDCOM file.

CONC {CONCATENATION}:=

Additional text for the superior line value. A CONC record without a line value is a fatal syntax error.

The CONC record exists to allow logical line values longer than the maximum physical line value. A GEDCOM writer must split long line values into parts and use subordinate CONC records to record the additional parts. A GEDCOM reader must append CONC line value must be appended to logical line value it is reconstituting for the superior record.
The CONC & CONT section discusses CONC & CONT usage in detail.

CONT {CONTINUATION}:=

A newline and usually some additional text as well for the superior line value. A CONT record without a line value is *not* an error, but the GEDCOM encoding of an empty line.

The CONT record exists to allow logical line values to contain newlines, while physical lines must not contain newlines. A GEDCOM writer must split a line values at a newline, and then continue writing the value using a subordinate CONT record. A GEDCOM reader must append a newline to the logical line value it is reconstituting for the superior record

before appending the CONT line value itself.
The CONC & CONT section discusses CONC & CONT usage in detail.

FORM {FORM}:=
The name of the *GEDCOM form* used in this GEDCOM file.
The GEDCOM specification defines just one GEDCOM form: `LINEAGE-LINKED`.

GEDC {GEDCOM}:=
Information about the use of GEDCOM in a GEDCOM file.

HEAD {HEADER}:=
Information about the entire GEDCOM file.

TRLR {TRAILER}:=
A top-level record that marks the end of the GEDCOM file.

VERS {VERSION}:=
Version number.

Chapter 2 Lineage-Linked Form

Introduction

The Lineage-Linked Form defined in this chapter is a GEDCOM form based on the general framework of the Basic GEDCOM Language defined in Chapter 1.

The Lineage-Linked Form is a couple-centric GEDCOM form; the family group record for a couple and their children is placed centrally in this design. This GEDCOM form is called lineage-linked because it pertains to individuals linked in parent-child relationships across multiple generations.
This chapter describes the specific tag, value, and pointer combinations used for exchanging genealogical information in this format. The chapter also addresses specific compatibility issues pertaining to previous Lineage-Linked Form releases and contains a simple lineage-linked GEDCOM file example.

Organisation

The basic description of the **Lineage-Linked GEDCOM Form** is presented in the following four major sections:

- "Lineage-Linked Form definition", page 55
- "Top-Level Records of the Lineage-Linked Form", page 56
- "Subrecords of the Lineage-Linked Form, page 64
- "Primitive elements of the Lineage-Linked Form", page 75

The definition of the tags used in defining the lineage-linked structures are contained in Appendix A.

Lineage-Linked Form Usage Conventions

- The order in which GEDCOM lines are written to a GEDCOM file is controlled by the context (level and parent record). The Lineage-Linked syntax may demand that particular records be in a particular order, e.g. the submitter record must follow directly after the GEDCOM header, but otherwise, the order of different records within the same context is not significant.
  It is possible for the same record type to occur multiple time within the same context. This happens in two situations: events that may happen more than once (e.g. burial), and events that happen just once, but for which there are multiple possible conflicting values (e.g. birth date). In the latter case, the order of the records is interpreted as the submitter's preference. *The most preferred value being the first with the least preferred data listed in subsequent lines by order of decreasing preference.* For example, a researcher who discovers conflicting evidence about a person's birth event would list the most credible information first and the least credible or least preferred items last.

- Systems that support multiple fields or structures should allow their users to indicate their preference opinion. Systems that only store single value structures should use the preferred information (the first occurrence listed) and store the remaining information as an exception, preferably *within an appropriate NOTE field* or in some way that the user has ready access to the less-preferred data when viewing the record.

- Conflicting event dates and places should be represented by placing them in separate event structures with appropriate source citations rather than by placing them under the same enclosing event.

- The Lineage-Linked GEDCOM Form uses the TYPE tag to classify its superior tag for the viewer. The value portion given by the TYPE tag is not intended to inform a computer program how to process the data, unless there is a list of standardised or controlled line_value choices given by the definition of the line value in this standard. The difference between an uncontrolled line value and a note value is that displaying systems should always display the *type value* when they display the data from the associated context. This gives the

user flexibility in further defining information in a compatible GEDCOM context and the reader to understand events or facts which have not been classified by a specific tag. For example:

1 EVEN
2 TYPE Awarded BSA Eagle Rank
2 DATE 1980

! All controlled line_value choices (enumerated values) are case-insensitive. The values “Feb”, “FEB” and “feb” are considered equal.
A GEDCOM reader must convert all such values to all-uppercase or all-lowercase prior to comparing to internally defined values.

GEDCOM tags are case-sensitive. All the tags of the Lineage-Linked Form are UPPERCASE.
User-defined tags should be UPPERCASE as well.

! All GEDCOM lines but the last one have either a **value** or a **pointer** unless the line contains subordinate GEDCOM lines. The Lineage-linked form **does not allow** a GEDCOM line with both a value and a pointer on the same line.

For anything but the TRLR record, the presence of only a level number and a tag, without any value or any subrecords, is a syntax error. For example, the way to assert that a death happened at an unknown time and place is not the line **1 DEAT** without subrecords, but the line **1 DEAT Y** without subrecords.

User-defined Records

Any record that isn't part of the Basic GEDCOM Language or the GEDCOM form used, is assumed to be a user-defined record. User-defined records make use of user-defined tags. All user-defined tags must start with an underscore (_).
Tags may not contain additional underscores. Developers who use tags with additional underscores in GEDCOM 5.5.1 or earlier may continue to do so, but must leave out these underscores in GEDCOM 5.5.5 or later.

additional rules regarding predefined tags

- It is illegal to use any user-defined tags when predefined tags are sufficient.
- It is illegal to define a user-defined tags that equals a predefined tag with an underscore in front.
- It is also illegal to try and “bring back” a tag that has been obsoleted in GEDCOM 5.5.5 or later by prefixing it with an underscore.
- It is legal to support a new record in earlier versions of GEDCOM by using the new tag prefixed with an underscore, but only in support of that new record in older versions.

So, it is illegal to use `_TOWN` to record the city within an address, because we already have the CITY record to do so. It is illegal to use `_CITY` for anything, because the `CITY` tag already exists. It is illegal to `_SSN` in GEDCOM 5.5.5 or later, because GEDCOM 5.5.5 obsoleted SSN. It is legal to use `_EMAIL` in GEDCOM 5.5, to support the EMAIL record introduced in GEDCOM 5.5.1.
The list of regular tags that may not be turned into user-defined by prefixing them with an underscore consists of all the predefined tags, and the tags obsoleted in GEDCOM 5.5.5.

End User-Defined Tags

There are developer-defined records and truly end user-defined records. The GEDCOM specification does not distinguish between those two categories. When the GEDCOM specification mentions user-defined records or tags, that generally means developer- and product-specific records and tags, but it is not uncommon for genealogy applications to allow users to define their own records and tags. It is up to applications to keep all user-defined tags legal by

making sure they all start with an underscore. Applications should keep all developer-defined tags all-uppercase, and use all-lowercase for truly end-user defined tags. This allows third parties to easily distinguish between these two different categories of so-called user-defined tags.

tag interpretation

The interpretation of developer-defined tags depends on the GEDCOM dialect as specified by HEAD.DEST. The interpretation of truly *end user*-defined tags is anyone's guess.

The HEAD.SOUR line value identifies the product that created the GEDCOM file - and that is all it identifies. The GEDCOM dialect used within a GEDCOM file, and thus the interpretation of GEDCOM extensions, is indicated by the HEAD.DEST line value.
In a typical GEDCOM file, the HEAD.DEST is equal to the HEAD.SOUR value.

Reading Third-Party Extensions

Although it is perfectly legal for any genealogy application to ignore so-called user-defined tags, that generally isn't what users want or expect. In practice, many genealogy software developers try to support the most important GEDCOM extensions used by other developers on GEDCOM import, as that provides a better user experience than ignoring them. Genealogy software developers with products that understand GEDCOM extensions used by other products often tout the product's ability to import GEDCOM files created by those products as a selling point.

Developers can and should improve third-party support for their extensions by publicly documenting them.

References

- GEDCOM SOUR and DEST
- GEDCOM System Identifiers
- Behold blog 2011-11-21: A Plethora of Extra GEDCOM Tags

Lineage-Linked Form Definition

The Lineage-Linked Form is a GEDCOM form. A file using the Lineage-Linked Form is known as a lineage-linked GEDCOM file.

A Lineage-Linked GEDCOM File

- is identified by the <<**GEDCOM_FORM**>> value `LINEAGE-LINKED`
- extends the Basic GEDCOM header with the <<**LINEAGE_LINKED_HEADER_EXTENSION**>>
- contains zero or more <<**LINEAGE_LINKED_RECORD**>>
- contains exactly one <<**SUBMITTER_RECORD**>> that follows directly after the GEDCOM header

<<**GEDCOM_FORM**>>

The definition of the Lineage-Linked Form builds on the definitions of the Basic GEDCOM Language. The high-level Lineage-Linked Form definitions are as follows:

GEDCOM_FORM:= **{Size=14:20}**

[LINEAGE-LINKED]
The line value `LINEAGE-LINKED` identifies the Lineage-Linked form.

A GEDCOM 5.5.5 reader *must* check the specified <GEDCOM_FORM> for validity. A

GEDCOM 5.5.5 reader must reject any file without a <GEDCOM_FORM> as not a GEDCOM file. If the <GEDCOM_FORM> isn't LINEAGE-LINKED or another supported GEDCOM form, the GEDCOM reader *must not* try to process the file, but must refuse to process the file with an *unsupported GEDCOM form* message.
Many GEDCOM 5.5 and GEDCOM 5.5.1 readers correct for known `LINEAGE-LINKED` spelling errors known to occur in GEDCOM 5.5 and 5.5.1 files. A GEDCOM 5.5.5 reader *need not* and *must not* do so.

GEDCOM_FORM_HEADER_EXTENSION:= **{1:1}**

n <<LINEAGE_LINKED_HEADER_EXTENSION>>	{1:1}	p. 57

HEAD subrecords specific to the Lineage-Linked Form.

FORM_RECORDS:=

0 <<SUBMITTER_RECORD>>	{1:1}	p. 63
0 <<LINEAGE_LINKED_RECORD>>	{0:M}	p. 58

A Lineage-Linked files consists of zero or more lineage-linked records, preceded by a submitter record. Thus, the submitter record follows directly after the GEDCOM header.

Lineage-Linked GEDCOM File

When you merge the <<GEDCOM_HEADER>> and the high-level Lineage-Linked Form definitions into the <<**GEDCOM_FILE**>> definition, you get the following Lineage-Linked GEDCOM file definition.

LINEAGE_LINKED_GEDCOM_FILE:=

U+FEFF (Byte Order Mark)		
n HEAD	{1:1}	
+1 **GEDC**	{1:1}	
+2 **VERS** <GEDCOM_VERSION_NUMBER>	{1:1}	p. 49
+2 **FORM** LINEAGE-LINKED	{1:1}	
+3 **VERS** <GEDCOM_VERSION_NUMBER>	{1:1}	p. 49
+1 **CHAR** <CHARACTER_ENCODING>	{1:1}	p. 47
+1 <<LINEAGE_LINKED_HEADER_EXTENSION>>	{1:1}	p. 57
0 <<SUBMITTER_RECORD>>	{1:1}	p. 63
0 <<LINEAGE_LINKED_RECORD>>	{0:M}	p. 58
0 <<GEDCOM_TRAILER>>	{1:1}	p. 51

Top-Level Records of the Lineage-Linked Form

LINEAGE_LINKED_GEDCOM_FILE:=

U+FEFF (Byte Order Mark)		
n HEAD	{1:1}	
+1 **GEDC**	{1:1}	
+2 **VERS** <GEDCOM_VERSION_NUMBER>	{1:1}	p. 49
+2 **FORM** LINEAGE-LINKED	{1:1}	
+3 **VERS** <GEDCOM_VERSION_NUMBER>	{1:1}	p. 49
+1 **CHAR** <CHARACTER_ENCODING>	{1:1}	p. 47
+1 <<LINEAGE_LINKED_HEADER_EXTENSION>>	{1:1}	p. 57
0 <<SUBMITTER_RECORD>>	{1:1}	p. 63
0 <<LINEAGE_LINKED_RECORD>>	{0:M}	p. 58
0 <<GEDCOM_TRAILER>>	{1:1}	p. 51

Submitter Record

A Lineage-Linked GEDCOM file consist of a header record, a trailer record, and other records in between. This top-level definition of the lineage-linked form includes the mandatory submitter record explicitly because it must follow the header record directly. This demand is made to avoid breaking existing GEDCOM parsers, which expect the submitter record to be there.

Version Number

For Lineage-Linked GEDCOM files based on the GEDCOM 5.5.5 specification, the GEDCOM version is `5.5.5` and the Lineage-Linked Form version is `5.5.5` too.

LINEAGE_LINKED_HEADER_EXTENSION:=

n **DEST**<RECEIVING_SYSTEM_NAME>	{0:1}	p. 103
n **SOUR** <SYSTEM_ID>	{1:1}	p. 106
+1 VERS <PRODUCT_VERSION_NUMBER>	{0:1}	p. 102
+1 NAME <NAME_OF_PRODUCT>	{0:1}	p. 96
+1 CORP <NAME_OF_BUSINESS>	{0:1}	p. 96
+2 <<ADDRESS_STRUCTURE>>	{0:1}	p. 64
+1 DATA <NAME_OF_SOURCE_DATA>	{0:1}	p. 96
+2 DATE <PUBLICATION_DATE>	{0:1)	p. 103
+2 COPR <COPYRIGHT_SOURCE_DATA>	{0:1)	p. 79
n DATE <FILE_CREATION_DATE>	{0:1}	p. 93
+1 TIME <TIME_VALUE>	{0:1}	p. 107
n LANG <LANGUAGE_OF_TEXT>	{0:1}	p. 94
n **SUBM** <XREF:SUBM>	{0:1}	p. 108
n FILE <GEDCOM_FILE_NAME>	{0:1}	p. 93
n COPR <COPYRIGHT_GEDCOM_FILE>	{0:1}	p. 79
n NOTE <GEDCOM_CONTENT_DESCRIPTION>	{0:1}	p. 93

Form Header

A GEDCOM form does not define a GEDCOM header, it merely defines a form-specific extension to the mandatory <GEDCOM_HEADER>.

Extensions

The Basic GEDCOM header *must not* contain CONC or CONT records.
A form header *may* contain CONC or CONT records, but this is strongly discommended.

The Basic GEDCOM header *must not* contain GEDCOM extensions. A form header *may* contain GEDCOM extensions, but this is strongly discommended.
The form header is complex enough as it is. Developers ignoring this caution should not expect third parties to support their extensions.

Valid Header

It is important that GEDCOM 5.5.5 writers produce correct GEDCOM files, and especially important that each file contains a correct form header. **A GEDCOM 5.5.5 reader must demand a flawless form header; it *must not* process a GEDCOM file if the header is invalid, but must issue a fatal error and abort.**

HEAD.SUBM Optional

GEDCOM 5.5.5 allows just one SUBM record per GEDCOM file, *and* it has to follow immediately after the GEDCOM header. That makes HEAD.SUBM record superfluous.

The HEAD.SUBM is no longer mandatory, but optional now. It may be deprecated in a later version. It is recommended that GEDCOM 5.5.5 writers do create the HEAD.SUBM record, for maximum compatibility with GEDCOM 5.5.1. The HEAD.SUBM pointer is not needed, but when the record is included, the pointer must be valid.

SOUR & DEST

The header structure provides information about the entire GEDCOM file. The SOUR (source) system name identifies which system created the file, and that is all it identifies.
The DEST (destination) system name identifies the intended receiving system.
The HEAD.DEST value identifies the GEDCOM dialects used, i.e. the interpretation of the GEDCOM extension. GEDCOM writers must default the HEAD.DEST value to be equal to the HEAD.SOUR value, to ensure correct interpretation of their system's GEDCOM extensions.

LINEAGE_LINKED_RECORD:=

```
[
n <<FAM_GROUP_RECORD>>                    {1:1}    p. 58
|
n <<INDIVIDUAL_RECORD>>                   {1:1}    p. 61
|
n <<MULTIMEDIA_RECORD>>                   {1:1}    p. 62
|
n <<NOTE_RECORD>>                         {1:1}    p. 63
|
n <<REPOSITORY_RECORD>>                   {1:1}    p. 63
|
n <<SOURCE_RECORD>>                       {1:1}    p. 63
]
```

User-Defined Top-Level Records

GEDCOM allows genealogy software developers to define additional records types, and that includes top-level record types.
For example, multiple developers use a top-level _PLACE or _LOC records to deal with GEDCOM 5.5.1's *place record design error*.

Bold Letters

Most genealogy applications generate cross-reference identifiers that consist of a single capital letter followed by a number, e.g. `@S123@` or `@F456@`. The emboldened letter for each type in the definition above is the suggested letter for records of that type.
A GEDCOM 5.5.5 file contains exactly one <<**S**UBMITTER_RECORD>>, it is suggested that its identifier be `@U1@`.

> **Reference**
>
> Common GEDCOM Identifier Naming Convention

FAM_GROUP_RECORD:=

```
n <XREF:FAM> FAM                          {1:1}
   +1 <<FAMILY_EVENT_STRUCTURE>>          {0:M}    p. 67
   +1 HUSB <XREF:INDI>                    {0:1}    p. 108
   +1 WIFE <XREF:INDI>                    {0:1}    p. 108
   +1 CHIL <XREF:INDI>                    {0:M}    p. 108
```

+1 NCHI <COUNT_OF_CHILDREN>	{0:1}	p. 79
+1 REFN <USER_REFERENCE_NUMBER>	{0:M}	p. 107
+2 TYPE <USER_REFERENCE_TYPE>	{0:1}	p. 107
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	p. 78
+1 <<CHANGE_DATE>>	{0:1}	p. 66
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71
+1 <<SOURCE_CITATION>>	{0:M}	p. 73
+1 <<MULTIMEDIA_LINK>>	{0:M}	p. 71

The FAM ("family group") record records a *family group*: a couple and their children, if any. The family group record is used to record marriages, common law marriages and other relationships, and all children from that relationship.

relationships

The FAM.MARR record documents the relationship between FAM.HUSB or FAM.WIFE. The nature of the relationship is documented by the optional MARR.TYPE subrecord; if there is no MARR.TYPE record, marriage is assumed.
The following table is a list of the common MARR.TYPE values and their meaning. Developers should use these values for maximum compatability with other applications, and must not use translations, abbreviations or other values that mean the same thing as the values documented here.

unknown	relationship (type unknown)
marriage	marriage
not married	not married
civil	civil marriage
religious	religious marriage
common law	common law marriage
partnership	partnership
registered partnership	registered partnership
living together	living together
living apart together	living apart together

The religious marriage ceremony that may follow a civil marriage is not included in this list, as it does not establish a relationship; it must be recorded as an event.
It would make sense for `unknown` to be the default; `marriage` is the default for backward compatability.

Lineage-Linked

In Lineage-Linked GEDCOM, parents and children are not linked directly to each other, but instead linked through a family group record. A child is linked to its known parent or parents via their family group record, and each family group record links to all known children for those parents.
The preferred order of the CHIL (children) pointers within a FAM (family group) structure is chronological by birth.

The family group record documents unions between at most two people. GEDCOM does not offer any record to document unions of three or more people. Multiple unions and polygamy can and must be recorded by using multiple family group records.

Family Group Record

Each family group record documents just one union between at most two people (one or both may be unknown). Each union, each relationship requires its own family group record. An individual who had more than one relationship, will occur in more than one family group record.
The family group record is the only record for documenting couples, and must be used for all couples.

The HUSB subrecord and WIFE subrecord link to the individuals in the relationship. Despite the gendered names of these subrecords, the family group record can and must be used to document same-sex relationships. It is legal to link the FAM.HUSB record to a female individual, or the FAM.WIFE record to a male individual.

Sex

That the names of HUSB & WIFE subrecords suggest a particular sex and gendered roles is an artefact of GEDCOM's history. The names of these subrecords remain unchanged from previous versions of GEDCOM for backward compatibility, but they should largely be thought of as awkward synonyms for a sex-neutral FAM.SPOU records.
The HUSB & WIFE subrecords do not indicate any sex, gender or role other than partner or parent. The FAM.HUSB will often link to a male individual, and FAM.WIFE will often link to a female individual, but that need not be the case.
Application *must not* assume any particular sex, gender or role based on the FAM.HUSB or FAM.WIFE subrecords.

Sex is specified by INDI.SEX and by INDI.SEX only. Any gendered pronouns used on screen or in reports must be based on the sex documented in the INDI record.

GEDCOM writers must avoid confusing other applications that do still make assumptions about gender or role based on the FAM.HUSB or FAM.WIFE subrecords.
If only a father is known, that father must be linked through the FAM.HUSB record. If only a mother is known, that mother must be linked through the FAM.WIFE record. For any male & female couple, FAM.HUSB must link to the male individual and FAM.WIFE to the female individual. Having the FAM.HUSB record point to the wife, and the FAM.WIFE record to the husband is an error.

Empty Records

There can be at most one HUSB subrecord and one WIFE subrecord per family group record. Neither one is mandatory. Either one can be missing to document one known and one unknown parent for a child of that couple. Both can be missing to document that we know children to be siblings, but do not know who their parents are.
The syntax even allows a family group record without either parents or children. Genealogy applications should not create “empty” family group records, but should automatically delete a family group record when a user has removed all links to individuals. A family group record for just one child makes no sense, but a family group for only a single partner should be maintained, if the MARR subrecord contains information, such as a marriage date (marriage date known, partner unknown).

Remarriage

There may be multiple events for a single relationship, but there may be just one relationship per FAM record. There may be just one relationship per family group record. *Every* new relationship requires a new family group record, even if that relationship is between two people who already had a previous relationship.
This consistent approach keeps things simple and allows other relationships in between.

The family group record allows multiple events of the same type, but they must be for same relationship. This provides the flexibility to record multiple marriage records with different dates and places for the same marriage.
There may be multiple marriages for the same relationship when the legality of one marriage is doubtful or rejected by another jurisdiction. There may also be multiple marriage records, with different dates and places, for a single marriage, for example when a Dutch couple marries in Germany and then goes back to their home town in the Netherlands, the German town will create a marriage record, and the Dutch town will create a marriage record several days or weeks later, when the couple comes back and shows the town clerk that they've been married.

Order of Children

Children within a family group should be ordered chronologically, i.e. in order of birth, and not anti-chronologically, and the order of multiplets born on the same day should be preserved.

Although it would certainly make some sense to let applications put children in order, it is current practice to let the user decide the order of children. Users expect genealogy applications to respect the order they choose, and not change it on import on export.

Genealogy Application Best Practice

- encourage users to enter dates, including approximate dates
- Encourage users to put and keep children in order:
  - when you show children, show their birth (or baptism) dates
  - visually indicate problems with the order of children
  - alert users when they insert a child out of order
  - include a children-in-order check as a basic genealogy consistency check
- Preserve the order of children across GEDCOM import and export
- optionally: make ordering children easy with a reorder children function
- checking chronological order
  - *when checking*, treat dates consisting of just a year as January 1 of that year
  - *when checking*, treat dates consisting of just a year and a month as the first day of that month
  - for multiplets born on the same day, simply respect the order chosen by the user

References

- Marriage in GEDCOM
- Married, Divorce, Married Again
- Same-Sex Marriage in GEDCOM

INDIVIDUAL_RECORD:=

n <XREF:INDI> **INDI**	{1:1}	
+1 <<PERSONAL_NAME_STRUCTURE>>	{0:M}	p. 72
+1 SEX <SEX_VALUE>	{0:1}	p. 105
+1 <<INDIVIDUAL_EVENT_STRUCTURE>>	{0:M}	p. 69
+1 <<INDIVIDUAL_ATTRIBUTE_STRUCTURE>>	{0:M}	p. 68
+1 <<CHILD_TO_FAMILY_LINK>>	{0:M}	p. 67
+1 <<SPOUSE_TO_FAMILY_LINK>>	{0:M}	p. 75
+1 <<ASSOCIATION_STRUCTURE>>	{0:M}	p. 65
+1 REFN <USER_REFERENCE_NUMBER>	{0:M}	p. 107
+2 TYPE <USER_REFERENCE_TYPE>	{0:1}	p. 107
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	p. 78
+1 <<CHANGE_DATE>>	{0:1}	p. 66
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71
+1 <<SOURCE_CITATION>>	{0:M}	p. 73
+1 <<MULTIMEDIA_LINK>>	{0:M}	p. 71

The individual record is a compilation of facts, known or discovered, about an individual. Sometimes these facts are from different sources. This form allows documentation of the source where each of the facts were discovered.

The GEDCOM 5.5.5 INDI is significantly simpler than the GEDCOM 5.5.1 INDI record; All the GEDCOM 5.5.1 subrecords that were marked obsolete or deprecated in the *GEDCOM 5.5.1 Annotated Edition* have been removed.

The INDI.SEX record is optional. When absent, systems *must* assume the default <SEX_VALUE> of `U`.

The normal lineage links are shown through the use of pointers from the individual to a family group through either the FAMC tag or the FAMS tag. The FAMC tag provides a pointer to a family group record where this person is a child. The FAMS tag provides a pointer to a family group record where this person is a spouse or parent. The <<**CHILD_TO_FAMILY_LINK**>>, page 67 structure contains a FAMC pointer which is required to show any child to parent linkage for pedigree navigation. The <<CHILD_TO_FAMILY_LINK>> structure also indicates whether the pedigree link represents an official lineage (birth record), or an adoption lineage.

Linkage between a child and the family group they belonged to at the time of an event can also be shown by a FAMC pointer subordinate to the appropriate event. For example, a FAMC pointer subordinate to an adoption event indicates a relationship to a family group by adoption. Official parents can be shown by a FAMC pointer subordinate to the birth event (optional).

Other associations or relationships are represented by the <<**ASSOCIATION_STRUCTURE**>> (the ASSO record).

Order of Parents

The INDI record may contain more than one FAMC subrecord, with each INDI.FAMC record pointing to a family group the individual was part of at sometime. These should be listed chronologically; thus, the official parents should be listed first, and the current legal parents should be listed last.

References

Behold blog: Sex in GEDCOM

MULTIMEDIA_RECORD:=

n <XREF:OBJE> **OBJE**	{1:1}	
+1 **FILE** <MULTIMEDIA_FILE_REFERENCE>	{1:1}	p. 95
+2 **FORM** <MULTIMEDIA_FORMAT>	{1:1}	p. 95
+3 TYPE <SOURCE_MEDIA_TYPE>	{0:1}	p. 106
+2 TITL <DESCRIPTIVE_TITLE>	{0:1}	p. 89
+1 REFN <USER_REFERENCE_NUMBER>	{0:M}	p. 107
+2 TYPE <USER_REFERENCE_TYPE>	{0:1}	p. 107
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	p. 78
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71
+1 <<SOURCE_CITATION>>	{0:M}	p. 73
+1 <<CHANGE_DATE>>	{0:1}	p. 66

The GEDCOM 5.5.1 <<MULTIMEDIA_RECORD>> allowed a single OBJE record to link to multiple related files.
That GEDCOM 5.5.1 feature is not widely supported; many genealogy applications will read only the first OBJE.FILE, resulting in loss of data. It was deprecated in the GEDCOM 5.5.1 Annotated Edition.
GEDCOM 5.5.5 simplifies the multimedia support by no longer allowing multiple files per media record. There is one multimedia record per file, one file per multimedia record.

GEDCOM does not include any grouping or tagging mechanism for multimedia files.

GEDCOM does not define a standard for bundling multimedia files with a GEDCOM file.

NOTE_RECORD:=

n <XREF:NOTE> **NOTE** <USER_TEXT>	{1:1}	p. 108
+1 REFN <USER_REFERENCE_NUMBER>	{0:M}	p. 107
+2 TYPE <USER_REFERENCE_TYPE>	{0:1}	p. 107
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	p. 78
+1 <<SOURCE_CITATION>>	{0:M}	p. 73
+1 <<CHANGE_DATE>>	{0:1}	p. 66

REPOSITORY_RECORD:=

n <XREF:REPO> **REPO**	{1:1}	
+1 **NAME** <NAME_OF_REPOSITORY>	{1:1}	p. 96
+1 <<ADDRESS_STRUCTURE>>	{0:1}	p. 64
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71
+1 REFN <USER_REFERENCE_NUMBER>	{0:M}	p. 107
+2 TYPE <USER_REFERENCE_TYPE>	{0:1}	p. 107
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	p. 78
+1 <<CHANGE_DATE>>	{0:1}	p. 66

SOURCE_RECORD:=

n <XREF:SOUR> **SOUR**	{1:1}	
+1 DATA	{0:1}	
+2 EVEN <EVENTS_RECORDED>	{0:M}	p. 92
+3 DATE <DATE_PERIOD>	{0:1}	p. 85
+3 PLAC <SOURCE_JURISDICTION_PLACE>	{0:1}	p. 106
+2 AGNC <RESPONSIBLE_AGENCY>	{0:1}	p. 104
+2 <<NOTE_STRUCTURE>>	{0:M}	p. 71
+1 AUTH <SOURCE_ORIGINATOR>	{0:1}	p. 106
+1 TITL <SOURCE_DESCRIPTIVE_TITLE>	{0:1}	p. 105
+1 ABBR <SOURCE_FILED_BY_ENTRY>	{0:1}	p. 106
+1 PUBL <SOURCE_PUBLICATION_FACTS>	{0:1}	p. 106
+1 TEXT <TEXT_FROM_SOURCE>	{0:1}	p. 106
+1 <<SOURCE_REPOSITORY_CITATION>>	{0:M}	p. 74
+1 REFN <USER_REFERENCE_NUMBER>	{0:M}	p. 107
+2 TYPE <USER_REFERENCE_TYPE>	{0:1}	p. 107
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	p. 78
+1 <<CHANGE_DATE>>	{0:1}	p. 66
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71
+1 <<MULTIMEDIA_LINK>>	{0:M}	p. 71

Source records are used to provide a bibliographic description of the source cited. (See the <<SOURCE_CITATION>> structure, page 73, which contains the pointer to this source record.)

SUBMITTER_RECORD:=

n <XREF:SUBM> **SUBM**	{1:1}	
+1 NAME <SUBMITTER_NAME>	{1:1}	p. 106
+1 <<ADDRESS_STRUCTURE>>	{0:1}	p. 64
+1 <<MULTIMEDIA_LINK>>	{0:M}	p. 71
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71
+1 <<CHANGE_DATE>>	{0:1}	p. 66

The submitter record identifies an individual or organisation that created the GEDCOM file. All records in the GEDCOM file are understood to be submitted by the same individual or organisation.

A GEDCOM 5.5.5 file contains exactly one <<SUBMITTER_RECORD>>, directly after the <<HEADER>>. There is no need for the HEAD.SUBM pointer anymore, so it optional now.

Subrecords of the Lineage-Linked Form

ADDRESS_STRUCTURE:=

n **ADDR**	{1:1}	
+1 ADR1 <ADDRESS_LINE1>	{0:1}	p. 75
+1 ADR2 <ADDRESS_LINE2>	{0:1}	p. 75
+1 ADR3 <ADDRESS_LINE3>	{0:1}	p. 76
+1 CITY <ADDRESS_CITY>	{0:1}	p. 75
+1 STAE <ADDRESS_STATE>	{0:1}	p. 76
+1 POST <ADDRESS_POSTAL_CODE>	{0:1}	p. 76
+1 CTRY <ADDRESS_COUNTRY>	{0:1}	p. 75
n PHON <PHONE_NUMBER>	{0:3}	p. 99
n EMAIL <ADDRESS_EMAIL>	{0:3}	p. 75
n FAX <ADDRESS_FAX>	{0:3}	p. 75
n WWW <ADDRESS_WEB_PAGE>	{0:3}	p. 76

History

The GEDCOM 5.5.1 specification supports both new style structured addresses (subrecords for separate address parts) and old style unstructured addresses (a single line value with line breaks). Structured addresses were introduced in GEDCOM 5.4 (1995), although the ADR3 tag was only added in GEDCOM 5.5.1 (1999).
The new style structured addresses were introduced to replace the old style unstructured addresses. GEDCOM 5.5 (1996) understandably continued to allow the use of old style unstructured addresses for backward compatibility with applications that did not support GEDCOM 5.5 yet. GEDCOM 5.5.1 (1999) continued to allow the old style unstructured addresses. The GEDCOM 5.5.1 Annotated Edition (2018) explicitly deprecated the old style, unstructured addresses. GEDCOM 5.5.5 supports structured addresses exclusively. The ADDR line must not have a line value.

Structure

Notice that the <<ADDRESS_STRUCTURE>> isn't a single GEDCOM record, but one mandatory record and a few optional records.
The <<ADDRESS_STRUCTURE>> always includes the ADDR record, while the PHON, EMAIL, FAX and WWW records are optional.
Note that these optional records may only appear in combination with the mandatory ADDR record; on this point, GEDCOM 5.5.5 agrees with GEDCOM 5.5.1, not GEDCOM 5.5.

PHONE, EMAIL, FAX & WWW

The PHON, EMAIL, FAX, and WWW records are *not* subrecords of the ADDR record, nor does the GEDCOM specification define another record that all these records are a subrecord of. The PHON, EMAIL, FAX, and WWW records simply appear at the same level as the ADDR record.

Although the order of different records at the same level should not matter, it is strongly suggested that GEDCOM writers always put the mandatory ADDR record first, and any optional PHON, EMAIL, FAX and WWW records directly after that.

Home, Work and Mobile

The <<ADDRESS_STRUCTURE>> allows up to three PHON records, EMAIL, FAX & WWW records. This allows inclusion of say a home, work and mobile phone number, but GEDCOM does not provide a mechanism for indicating which phone number is which.

PAF Addresses use URL instead of WWW

FamilySearch PAF uses GEDCOM 5.5.1, but PAF addresses do not use the WWW tag specified here, they use the illegal tag URL instead. That is an error in PAF, and FamilySearch should have fixed PAF. Instead, FamilySearch "fixed" the GEDCOM specification: in GEDCOM 5.6, the tag has changed from WWW to URL.

Best Practices

GEDCOM Writer Best Practice

- Write the mandatory ADR record before any of the optional records
- Write any optional records directly after the mandatory ADR record

GEDCOM 5.5.1 Writer Best Practice

- Use structured addresses exclusively

GEDCOM 5.5.1 Reader Best Practice

- Accept URL as a synonym for WWW
- Do issue an non-fatal error stating that the URL tag is illegal

GEDCOM 5.5.5 Reader Best Practice

- Do not accept any nonsense

References

GEDCOM ADDR

ASSOCIATION_STRUCTURE:=

n **ASSO** <XREF:INDI>	{1:1}	p. 108
+1 **RELA** <RELATION_IS_DESCRIPTOR>	{1:1}	p. 104
+1 <<SOURCE_CITATION>>	{0:M}	p. 73
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71

The association pointer only associates INDI (individual) records to INDI (individual) records.

The ASSO record may only be used for relationships that aren't otherwise supported by the Lineage-linked Form. Parent-child relationship must be recorded using the INDI.FAMC and the FAM.CHIL records.

The person's relation or association is the person being pointed to. The association or relationship is stated by the value on the subordinate RELA line. For example:

0 @I1@ INDI
1 NAME Fred /Jones/
1 ASSO @I2@
2 RELA Godfather

This GEDCOM fragment states that `@I2@` is Fred's godfather.

The <RELATION_IS_DESCRIPTOR> does not provide a list of standard values. The ASSO.RELA line value is free-form text that has meaning to the user, not the application.

GEDCOM-L ASSO.RELA

The ASSO.RELA value should be assumed to be nothing but user-provided text, but it is worth noting that German developers working together through the GEDCOM-L mailing list have agreed to use the following values where appropriate:

- `Godparent`
- `Witness_of_Birth`
- `Witness_of_Death`

GEDCOM-L _ASSO.RELA

They also use a GEDCOM extension, the _ASSO record, for relationship from an individual to a couple, with the following agreed-upon _ASSO.RELA values:

- `Witness_of_Marriage`
- `Witness_of_Civil_Marriage`
- `Witness_of_Religious_Marriage`

Discommended

Earlier versions of GEDCOM featured support for witnesses through the WITN record. FamilySearch GEDCOM 5.4 eliminated the WITN record. Various genealogy software developers solve that limitation through GEDCOM extensions, most commonly through a _WITN record.
This ASSO and _ASSO records approach is strongly discommended.

Witnesses should not be associated with a couple. Witnesses must be associated with the event they witnessed. This _ASSO record approach fails to associate witnesses with an event and because of that, cannot even tell you which marriage someone witnessed.
It is better to use some product-specific _WITN record on the event itself.

References

GenWiki: GEDCOM/ASSO-Tag

Best Practice

- Applications should not assume any particular ASSO.RELA value has any particular meaning.
- Applications should issue a strong warning when an ASSO.RELA value seems to duplicate a familial relationship (e.g. grandfather)

CHANGE_DATE:=

n **CHAN**		{1:1}	
+1 **DATE** <DATE_EXACT>		{1:1}	p. 83
+2 TIME <TIME_VALUE>		{0:1}	p. 107
+1 <<NOTE_STRUCTURE>>		{0:M}	p. 71

The change date is intended to only record the last change to a record. Some systems may want to manage the change process with more detail, but it is sufficient for GEDCOM purposes to indicate the last time that a record was modified.

CHILD_TO_FAMILY_LINK:=

n **FAMC** <XREF:FAM>	{1:1}	p. 108
+1 PEDI <PEDIGREE_LINKAGE_TYPE>	{0:1}	p. 99
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71

EVENT_DETAIL:=

n TYPE <EVENT_OR_FACT_CLASSIFICATION>	{0:1}	p. 91
n DATE <DATE_VALUE>	{0:1}	p. 87
n <<PLACE_STRUCTURE>>	{0:1}	p. 72
n <<ADDRESS_STRUCTURE>>	{0:1}	p. 64
n AGNC <RESPONSIBLE_AGENCY>	{0:1}	p. 104
n RELI <RELIGIOUS_AFFILIATION>	{0:1}	p. 104
n CAUS <CAUSE_OF_EVENT>	{0:1}	p. 78
n <<NOTE_STRUCTURE>>	{0:M}	p. 71
n <<SOURCE_CITATION>>	{0:M}	p. 73
n <<MULTIMEDIA_LINK>>	{0:M}	p. 71

FAMILY_EVENT_DETAIL:=

n HUSB	{0:1}	
+1 AGE <AGE_AT_EVENT>	{1:1}	p. 77
n WIFE	{0:1}	
+1 AGE <AGE_AT_EVENT>	{1:1}	p. 77
n <<EVENT_DETAIL>>	{0:1}	p. 67

FAMILY_EVENT_STRUCTURE:=

[		
n [**ANUL** | **CENS** | **DIV** | **DIVF**]	{1:1}	
+1 <<FAMILY_EVENT_DETAIL>>	{0:1}	p. 0
|		
n [**ENGA** | **MARB** | **MARC**]	{1:1}	
+1 <<FAMILY_EVENT_DETAIL>>	{0:1}	p. 0
|		
n **MARR** [Y|<NULL>]	{1:1}	
+1 <<FAMILY_EVENT_DETAIL>>	{0:1}	p. 0
|		
n [**MARL** | **MARS**]	{1:1}	
+1 <<FAMILY_EVENT_DETAIL>>	{0:1}	p. 0
|		
n **RESI**	{1:1}	
+1 <<FAMILY_EVENT_DETAIL>>	{0:1}	p. 0
|		
n **EVEN** [<EVENT_DESCRIPTOR> | <NULL>]	{1:1}	p. 91
+1 <<FAMILY_EVENT_DETAIL>>	{0:1}	p. 0
]		

Residence isn't an Event but an Attribute

This <FAMILY_EVENT_STRUCTURE> definition includes RESI (residence) as a possible event. However, the residences of a family aren't events, they are attributes. A building has an age, but a residence has a period. Moving in and moving out are events, typically happening on a single day, while having residence is an attribute,

typical valid for a period of many years.

The miscategorisation of RESI as an event is a hold-over from earlier GEDCOM version. The FACT record was only introduced in GEDCOM 5.5.1. Prior to GEDCOM 5.5.1, FamilySearch did not really distinguish between events and attributes.

INDIVIDUAL_ATTRIBUTE_STRUCTURE:=

```
[
n CAST <CASTE_NAME>                                   {1:1}   p. 78
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n DSCR <PHYSICAL_DESCRIPTION>                         {1:1}   p. 100
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n EDUC <SCHOLASTIC_ACHIEVEMENT>                       {1:1}   p. 105
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n IDNO <ID_NUMBER>                                    {1:1}   p. 93
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {1:1}   p. 107
|
n NATI <NATIONAL_OR_TRIBAL_ORIGIN>                    {1:1}   p. 99
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n NCHI <COUNT_OF_CHILDREN>                            {1:1}   p. 79
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n NMR <NUMBER_OF_RELATONSHIPS>                        {1:1}   p. 99
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n OCCU <OCCUPATION>                                   {1:1}   p. 99
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n PROP <POSSESSIONS>                                  {1:1}   p. 102
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n RELI <RELIGIOUS_AFFILIATION>                        {1:1}   p. 104
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n RESI                                                {1:1}
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n TITL <NOBILITY_TYPE_TITLE>                          {1:1}   p. 99
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
   +1 TYPE <USER_REFERENCE_TYPE>                      {0:1}   p. 107
|
n FACT <ATTRIBUTE_DESCRIPTOR>                         {1:1}   p. 77
   +1 <<INDIVIDUAL_EVENT_DETAIL>>                     {0:1}   p. 69
```

+1 **TYPE** <USER_REFERENCE_TYPE> {1:1} p. 107
]

The IDNO and FACT record both *require* a subordinate TYPE record to identify what kind of number or fact is being recorded.
The TYPE record *may* be used with each of the above tags used in this structure.

INDIVIDUAL_EVENT_DETAIL:=

n <<**EVENT_DETAIL**>> {1:1} p. 67
n AGE <AGE_AT_EVENT> {0:1} p. 77

> **Attributes do not have Age**
>
> This FamilySearch GEDCOM 5.5.1 specification uses <INDIVIDUAL_EVENT_DETAIL> for both attributes and events. The inclusion of the AGE_AT_EVENT subrecord makes sense for events, but not for attributes.
> Some attributes (for example residence and occupation) have *periods* associated with them.
> The use of <INDIVIDUAL_EVENT_DETAIL> for attributes is a specification error, attributes should use a separate <INDIVIDUAL_ATTRIBUTE_DETAIL>.

INDIVIDUAL_EVENT_STRUCTURE:=

[
n **BIRT** {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
+1 FAMC <XREF:FAM> {0:1}
|
n **CHR** [Y | <NULL>] {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
+1 FAMC <XREF:FAM> {0:1}
|
n **DEAT** [Y | <NULL>] {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
|
n [**BURI** | **CREM**] {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
|
n **ADOP** {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
+1 FAMC <XREF:FAM> {0:1} p. 108
+2 ADOP <ADOPTED_BY_WHICH_PARENT> {0:1} p. 76
|
n [**BAPM** | **BARM** | **BASM**] {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
|
n [**CHRA** | **CONF** | **FCOM**] {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
|
n [**NATU** | **EMIG** | **IMMI**] {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
|
n [**CENS** | **PROB** | **WILL**] {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
|
n [**GRAD** | **RETI**] {1:1}
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
|

n **EVEN** [<EVENT_DESCRIPTOR> | <NULL>] {1:1} p. 91
+1 <<INDIVIDUAL_EVENT_DETAIL>> {0:1} p. 69
]

The GEDCOM 5.5.5 specification no longer allows the informationless GEDCOM record `1 BIRT Y`. The above definition includes the <EVENT_DESCRIPTOR> that was missing GEDCOM 5.5.1.

As a general rule, events are things that happen on a specific date. Use the date form 'BET date AND date ' to indicate that an event took place sometime between two dates. Resist the temptation to use a 'FROM date TO date ' form in an event structure. If the subject of your recording occurred over a period of time, then it is probably not an event, but rather an attribute or fact.

The EVEN tag in this structure is for recording general events that are not shown in the above <<INDIVIDUAL_EVENT_STRUCTURE>>. The event indicated by this general EVEN tag is defined by the value of the subordinate TYPE tag. For example, a person that signed a lease for land dated October 2, 1837 and a lease for equipment dated November 4, 1837 would be written in GEDCOM as:

1 EVEN
2 TYPE Land Lease
2 DATE 2 OCT 1837
1 EVEN
2 TYPE Equipment Lease
2 DATE 4 NOV 1837

The TYPE tag can be optionally used to modify the basic understanding of its superior event or attribute. For example:

1 GRAD
2 TYPE College

The occurrence of an event is asserted by the presence of either a DATE tag and value or a PLAC (place) tag and value in the event structure. When neither the date value nor the place value are known then a `Y` (yes) value on the parent event tag line is required to assert that the event happened. For example each of the following GEDCOM structures assert that a death happened:

1 DEAT Y
1 DEAT
2 DATE 2 OCT 1937
1 DEAT
2 PLAC Cove, Cache, Utah, United States of America

This convention asserts that the event happened.
It is not proper GEDCOM form to use a `N` (no) value with an event tag to record that it did not happen.

CHR Y, DEAT Y

The BIRT (birth), CHR (christening), and DEAT (death) dates may be unknown. The line value of these records may be empty (indicated by <NULL> in the definition), but, as stated in chapter 1, records must have value; the line value of most records may only be empty if it has subrecords. If there are no subrecords, the line value `Y` must be used.

The line `1 BIRT Y` means *born*; it provides no information whatsoever. The *GEDCOM 5.5.1 Annotated Edition* deprecated it. GEDCOM 5.5.5 makes it illegal.

The line `1 CHR Y` means *christened* (but do not know where or when), and the line `1 DEAT Y` means *died* (but do not know where or when).

MULTIMEDIA_LINK:=

n **OBJE** <XREF:OBJE> {1:1} p. 108

A <<MULTIMEDIA_LINK>> is a link to a <<MULTIMEDIA_RECORD>> (which is a link to a multimedia file).
There should be one multi-media record per file. There may be multiple multi-media links to a multi-media record.

The GEDCOM 5.5.1 specification demands support for the GEDCOM 5.5 <<MULTIMEDIA_LINK>>, because FamilySearch's own Personal Ancestral File (PAF) uses GEDCOM 5.5.1 yet uses the GEDCOM 5.5 <<MULTIMEDIA_LINK>>. That is a GEDCOM 5.5.1 specific issue. PAF does not create GEDCOM 5.5.5 files.
GEDCOM 5.5.5 does not tolerate the use of records defined in another GEDCOM version.

NOTE_STRUCTURE:=

[
n **NOTE** <XREF:NOTE> {1:1} p. 108
|
n **NOTE** <USER_TEXT> {1:1} p. 107
]

A NOTE line value will frequently be too long to fit on a single GEDCOM line, thus necessitating the use of CONC and CONT records.
GEDCOM writers and readers must follow the rules in the *CONC & CONT* section. The essence is that a GEDCOM reader must accept all GEDCOM lines as-is, *must not* strip leading or trailing white space. That simple rule prevent the loss of white space between words.

PERSONAL_NAME_PIECES:=

n NPFX <NAME_PIECE_PREFIX> {0:1} p. 97
n GIVN <NAME_PIECE_GIVEN> {0:1} p. 97
n NICK <NAME_PIECE_NICKNAME> {0:1} p. 97
n SPFX <NAME_PIECE_SURNAME_PREFIX {0:1} p. 98
n SURN <NAME_PIECE_SURNAME> {0:1} p. 98
n NSFX <NAME_PIECE_SUFFIX> {0:1} p. 98
n <<NOTE_STRUCTURE>> {0:M} p. 71
n <<SOURCE_CITATION>> {0:M} p. 73

Rufnamen

It is odd that the GEDCOM specification does not support call names. Many Americans have German ancestors and in Germany, people do not just have call names, their rufnamen (call names) are *official*, recorded in official documents. German genealogy software uses the _RUFNAME record to record these call names.

References

- GenWiki: GEDCOM/NAME-Tag

PERSONAL_NAME_STRUCTURE:=

n **NAME** <NAME_PERSONAL>	{1:1}	p. 96
+1 TYPE <NAME_TYPE>	{0:1}	p. 98
+1 <<PERSONAL_NAME_PIECES>>	{0:1}	p. 71
+1 FONE <NAME_PHONETIC>	{0:M}	p. 97
+2 TYPE <PHONETISATION_METHOD>	{1:1}	p. 100
+2 <<PERSONAL_NAME_PIECES>>	{0:1}	p. 71
+1 ROMN <NAME_ROMANISED>	{0:M}	p. 98
+2 TYPE <ROMANISATION_METHOD>	{1:1}	p. 105
+2 <<PERSONAL_NAME_PIECES>>	{1:1}	p. 71

The name value is formed in the manner the name is normally spoken, with the given name and family name (surname) separated by slashes (/). (See <NAME_PERSONAL>, page 96 for examples.) Based on the dynamic nature or unknown compositions of naming conventions, it is difficult to provide more detailed name piece structure to handle every case.

The NPFX, GIVN, NICK, SPFX, SURN, and NSFX tags are not optional.
Usage of the GEDCOM records for name parts is mandatory (for every name part that has a value).

Systems that still do not produce these name part records on export cannot use GEDCOM 5.5.5, but must stick to using GEDCOM 5.5.1 or earlier. A GEDCOM 5.5.5 reader expects these name parts, and uses these to correctly split names into their parts.

A GEDCOM 5.5.5 reader *must* split a name as specified by the GEDCOM file, *must not* split a name any other way. A GEDCOM 5.5.5 reader must check that each <<PERSONAL_NAME_PIECES>> (the NAME record and all its subrecords) is self-consistent, and must report a fatal error and abort import when it is not.

A <NAME_TYPE> is used to specify the particular variation that this name is. For example; if the name type is subordinate to <NAME_PERSONAL>, page 96 it could indicate that this name is a name taken at immigration or that it could be an 'also known as' name (see NAME_TYPE, page 98.)

PLACE_STRUCTURE:=

n **PLAC** <PLACE_NAME>	{1:1}	p. 100
+1 FONE <PLACE_PHONETIC>	{0:M}	p. 101
+2 TYPE <PHONETISATION_METHOD>	{1:1}	p. 100
+1 ROMN <PLACE_ROMANISED>	{0:M}	p. 101
+2 TYPE <ROMANISATION_METHOD>	{1:1}	p. 105
+1 MAP	{0:1}	
+2 LATI <PLACE_LATITUDE>	{1:1}	p. 100
+2 LONG <PLACE_LONGITUDE>	{1:1}	p. 100
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71

Place Record Design Error

The FamilySearch GEDCOM 5.5.1 specification added the place coordinates, the LATI and LONG records, as subrecords of the PLAC record. That sounds reasonable and logical, there is no better place (pun intended) to put these. However, the PLAC record is itself a subrecord, and a typical GEDCOM file contains the same place names over and over again.

The GEDCOM 5.5.1 design seems to expect that GEDCOM writers, when place coordinates are known, include those coordinates with each occurrence of each place name, thus bloating GEDCOM files even more than frequently repeated place names

already do.
Worse than that, GEDCOM readers that come across the same place name more than once, may come across conflicting coordinates, and cannot resolve that by keeping all the different ones; this PLAC record allows just one set of coordinates.

Top-Level Place Record

Genealogy applications developers have long desired the introduction of a top-level place record, and the introduction of place coordinates in GEDCOM 5.5.1 made a top-level place record a practical necessity, yet it was still not included.

Top-Level _PLAC Record

Several developers have addressed this design error by introducing a top-level _PLAC record, and do not record place coordinates within PLAC records, but only within _PLAC records.
Different genealogy software developers use different top-level _PLAC records, but still match PLAC record with those _PLAC records the same way: a PLAC and _PLAC record match each other if they contain exactly the same place name string.

Top-Level _LOC Record

Several German software developers created a collective set of extensions called GEDCOM EL (Extended Locations). The GEDCOM EL extension were originally created for GEDCOM 5.5, but are also used with GEDCOM 5.5.1.
GEDCOM EL uses a top-level _LOC record.

References

- Gaenovium Presentations: Louis Kessler: Reading Wrong GEDCOM Right
- Behold blog 2011 Dec 24: The Place Record in GEDCOM
- Detecting GEDCOM EL
- GenWiki: GEDCOM 5.5EL

SOURCE_CITATION:=

n **SOUR** <XREF:SOUR>	{1:1}	p. 108
+1 PAGE <WHERE_WITHIN_SOURCE>	{0:1}	p. 107
+1 EVEN <EVENT_TYPE_CITED_FROM>	{0:1}	p. 92
+2 ROLE <ROLE_IN_EVENT>	{0:1}	p. 104
+1 DATA	{0:1}	
+2 DATE <ENTRY_RECORDING_DATE>	{0:1}	p. 91
+2 TEXT <TEXT_FROM_SOURCE>	{0:M}	p. 106
+1 <<MULTIMEDIA_LINK>>	{0:M}	p. 71
+1 <<NOTE_STRUCTURE>>	{0:M}	p. 71
+1 QUAY <CERTAINTY_ASSESSMENT>	{0:1}	p. 78

The data provided in the <<SOURCE_CITATION>> structure is source-related information specific to the data being cited. (See *Sample Lineage-Linked GEDCOM File*, page 110 for an example.)

The information intended to be placed in the citation structure includes:

- The pointer to the <<SOURCE_RECORD>>, which contains a more general description of the source used for the fact being cited.
- Information, such as a page number, to help the user find the cited data within the referenced source. This is stored in the ".SOUR.PAGE" record.
- Actual text from the source that was used in making assertions, for example a date

phrase as actually recorded in the source, or significant notes written by the recorder, or an applicable sentence from a letter. This is stored in the ".SOUR.DATA.TEXT" record.

! Data that allows an assessment of the relative value of one source over another for making the recorded assertions (primary or secondary source, etc.). Data needed for this assessment is data that would help determine how much time from the date of the asserted fact and when the source was actually recorded, what type of event was cited, and what type of role did this person have in the cited source.
   - Date when the entry was recorded in source document is stored in the ".SOUR.DATA.DATE" record.
   - The type of event that initiated the recording is stored in the ".SOUR.EVEN" record. The value used is the event code taken from the table of choices shown in the <EVENT_TYPE_CITED_FROM> primitive, page 92.
   - The role of this person in the event is stored in the ".SOUR.EVEN.ROLE" record.

GEDCOM 5.4 (1995) introduced source citations using a reference to a (top-level) <<SOURCE_RECORD>> (SOUR record) and made it the preferred source citation format, to reduce redundancy. GEDCOM 5.5 and 5.5.1. continued to allow old format source citations that do not use SOUR record for backward compatibility with old systems, but clearly stated that the new format is preferred.
GEDCOM 5.5.5 no longer supports the old format, which has been deprecated since 1995.
In GEDCOM 5.5.5, each source citation *must* reference a (top-level) SOUR record.
Existing applications that still support the unstructured SOURCE_CITATION format should actively encourage users to upgrade all such citations (or at least the ones that cannot be mapped automatically), so that the user can take advantage of its GEDCOM 5.5.5 support.

SOURCE_REPOSITORY_CITATION:=

n **REPO** <XREF:REPO>	{1:1}	p. 108
+1 CALN <SOURCE_CALL_NUMBER>	{0:1}	p. 105
+2 MEDI <SOURCE_MEDIA_TYPE>	{0:1}	p. 106

The <<SOURCE_REPOSITORY_CITATION>> is an optional subrecord of the <<SOURCE_RECORD>>.
Applications should encourage users to always reference a repository for consulted sources, but it isn't necessary to identify a particular repository for a widely available published work.

The <<SOURCE_REPOSITORY_CITATION>> is used for both formal and informal repositories. An example of an informal repository is an uncle who possesses an old bible that contains genealogical notes; you would enter his name and address as the repository. For formal repositories, such as a library, the call number of the particular source may be recorded.

The GEDCOM 5.5.5 <<SOURCE_REPOSITORY_CITATION>> definition is simpler than the GEDCOM 5.5.1 definition, because GEDCOM 5.5.5 demands that all systems support the <<REPOSITORY_RECORD>> record.

FamilySearch deprecated the NOTE-only <<SOURCE_REPOSITORY_CITATION>> format 25 years ago (see <<SOURCE_CITATION>> annotation about new structured format replacing the old unstructured format).
Existing systems that still support the NOTE-only format must encourage users to change such NOTE into a REPO records, so that the user can take advantage of its GEDCOM 5.5.5 support.

This structure is used within a source record to point to a name and address record of the holder of the source document. Formal and informal repository names and addresses are stored in the <<REPOSITORY_RECORD>>. Informal repositories include owner's of an unpublished work or of a rare published source, or a keeper of personal collections. An example would be the owner of a family bible containing unpublished family genealogical entries. More formal repositories, such as the Family History Library, should show a call number of the source at that repository. The call number of that source should be recorded using a subordinate CALN tag. Systems which do not use repository name and address record, should describe where the information cited is stored in the

<<NOTE_STRUCTURE>> of the REPO (repository) source citation structure.

REPO.CALN.MEDI

There is a puzzling mistake that has, for the sake of full backward compatibility with (the not deprecated part of) the GEDCOM 5.5.1 structure, not been corrected in GEDCOM 5.5.5; the <<SOURCE_REPOSITORY_CITATION>> definition allows the CALN subrecord to have MEDI subrecord to specify a <<SOURCE_MEDIA_TYPE>>. That doesn't make sense.
The media type is a property of the source (<<SOURCE_RECORD>>), not of the repository, and certainly not of the call number.
Consider REPO.CALN.MEDI strongly deprecated.

SPOUSE_TO_FAMILY_LINK:=

n **FAMS** <XREF:FAM>		{1:1}	p. 108
+1 <<NOTE_STRUCTURE>>		{0:M}	p. 71

Primitive Elements of the Lineage-Linked Form

The field {Size=} specifications show the minimum and maximum field length. The field lengths are specified independent of other GEDCOM elements, such as level numbers and tags. GEDCOM lines are limited to 255 code units, but the GEDCOM grammar's CONC (concatenation) and CONT (continuation) records allow splitting a field over multiple lines.

ADDRESS_CITY:= **{Size=1:60}**

The name of the city used in the address. Isolated for sorting or indexing.

ADDRESS_COUNTRY:= **{Size=1:60}**

The name of the country that pertains to the associated address. Isolated by some systems for sorting or indexing. Used in most cases to facilitate automatic sorting of mail.

ADDRESS_EMAIL:= **{Size=5:120}**

An electronic address that can be used for contact such as an email address.

Email addresses contain a single at sign (@), but GEDCOM lines must not contain a single at sign.
In the GEDCOM grammar, a single at sign indicates the starts or end of an escape sequence. The GEDCOM grammar states that a single at sign must be encoded as a double at sign.

- A GEDCOM writer must export the single at sign as two consecutive at signs.
- A GEDCOM reader must import that double at sign as a single at sign.

ADDRESS_FAX:= **{Size=5:60}**

A FAX telephone number appropriate for sending data facsimiles.

ADDRESS_LINE1:= **{Size=1:60}**

The first line of the address used for indexing.

ADDRESS_LINE2:= **{Size=1:60}**

The second line of the address used for indexing.

ADDRESS_LINE3:= **{Size=1:60}**

The third line of the address used for indexing.

ADDRESS_POSTAL_CODE:= **{Size=1:10}**

The ZIP or postal code used by the various localities in handling of mail. Isolated for sorting or indexing.

ADDRESS_STATE:= **{Size=1:60}**

The name of the province, state or similar country subdivision used in the address. Isolated for sorting or indexing.

ADDRESS_WEB_PAGE:= **{Size=4:2047}**

The world wide web page address.

Minimum and Maximum URL Length

Minimum URL Length

The actual minimum URL length is two letters: just a top-level domain. The only known example is the now retired URL shortener `to`, which consist of nothing but the country code for Tonga. The practical minimum for a field like this four letters, a one-letter second level domain on a two-letter top-level domain, for example `t.co`, twitter's URL shortener.

Maximum URL Length

There is no official maximum URL length, but there are practical limitations. RFC7230 recommends that length of at least 8000 bytes should be supported by all servers and clients, but practical limits are lower. Firefox support longer ULS, but will not display more than the first 65.536 code units of URL in the address bar. A default Apache server configuration has a 8192-byte limit on individuals fields of a request. The maximum length for an URL in a sitemap protocol is 2048 code units. Internet Explorer has a maximum URL length of 2083 code units, and that value is provided in `WinINET.h` as `INTERNET_MAX_URL_LENGTH`, but the address bar has a limit of 2047 code units.

The real-world answer is that very long URLs are a mistake, but it is not up to the GEDCOM specification to unduly limit the length of URLs. That is why the max length has been set to 2047 code units.

References

- Quora: What's the real minimum URL length?
- IEInternals: URL Length Limits
- stack overflow: What is the maximum length of a URL in different browsers?

ADOPTED_BY_WHICH_PARENT:= **{Size=4:4}**

[HUSB | WIFE | BOTH]
A code which shows which parent in the associated family group record adopted this person.

where:

HUSB = The HUSB in the associated FAM record adopted this person.

WIFE = The WIFE in the associated FAM record adopted this person.
BOTH = Both HUSB and WIFE adopted this person.

The names of the FAM.HUSB and FAM.WIFE subrecords are historical. These subrecords identify individuals, they do *not* indicate particular roles within the family group.

AGE_AT_EVENT:= {Size=2:13}

[NULL
| < + space
| > + space]
]
[YYY + y + space + MM + m + space + DDD + d
| YYY + y
| MM + m
| DDD + d
| YYY + y + space + MM + m
| YYY+ y + space + DDD + d
| MM + m + space + DDD + d
| CHILD
| INFANT
| STILLBORN
]

where:

>	= greater than indicated age
<	= less than indicated age
y	= a label indicating years
m	= a label indicating months
d	= a label indicating days
YYY	= number of full years, at most three digits
MM	= number of months, at most 11, at most two digits
DDD	= number of days, at most 365, at most three digits
CHILD	= age < 8 years
INFANT	= age < 1 year
STILLBORN	= died just prior, at, or near birth, 0 years
space	= U+0020, the Space character

Notice that, in the above syntax, the uppercase letters represents a number, while the lowercase letters are to be included literally, as part of the line value.

A number that indicates the age in years, months, and days that the principal was at the time of the associated event. Any labels must come after their corresponding number, for example; 4y 8m 10d.

The line value should be normalised; it should for example not specify `2y 13m`, but `3y 1m` instead. Number of days is allowed to be 365 because of leap years.

The YYY, MM and DDD values must not be zero; if a value equals zero, that part is left off. The values may not contain leading zeroes either.

The values CHILD, INFANT, STILLBORN have been deprecated because they break the mold, are technically superflous, and not likely to be understood as precise as they are defined here. They continue to be allowed for compatability with GEDCOM 5.5.1 only.

A notable shortcoming of the AGE_AT_EVENT syntax it that it does not support hours or minutes. Recording that a child died minutes or hours after birth is not supported.

ATTRIBUTE_DESCRIPTOR:= {Size=1:90}

Text describing a particular characteristic or attribute assigned to an individual. This attribute value is assigned to the FACT record. The classification of this specific attribute or fact is specified by the value of the subordinate TYPE record selected from the

<EVENT_DETAIL> structure. For example if you were classifying the skills a person had obtained;

1 FACT Woodworking
2 TYPE Skills

ATTRIBUTE_TYPE:= **{Size=4:4}**

[CAST | EDUC | NATI | OCCU | PROP | RELI | RESI | TITL | FACT]
An attribute which may have caused name, addresses, phone numbers, family listings to be recorded. Its application is in helping to classify sources used for information.

AUTOMATED_RECORD_ID:= **{Size=1:12}**

A unique record identification number assigned to the record by the source system. This number is intended to serve as a more sure means of identification of a record for reconciling differences in data between two interfacing systems.

BEFORE_COMMON_ERA:= **{Size=2:4}**

[BCE | BC | B.C.]

<BEFORE_COMMON_ERA> allows three different values, all meaning the same thing. A GEDCOM 5.5.5 reader must support all three.
While enumerated values are case-insensitive, it is recommended that GEDCOM writers always use all-uppercase for those values, as that is what human readers examining GEDCOM files expect.

GEDCOM 5.5.1 specifies the use of `B.C.` (two full stops), but FamilySearch's own Personal Ancestral File (PAF) uses `BC` (just two letters) instead, and so do other applications. Therefore, the *GEDCOM 5.5.1 Annotated Edition* recommends that GEDCOM readers accept both values, *and* `BCE` as well.

A GEDCOM 5.5.5 reader cannot simply continue that GEDCOM 5.5.1 best practice, as a GEDCOM 5.5.5 reader *must not* accept invalid values. To continue to allow acceptance of both `B.C.` and `BC`, both are defined as legal. Both are deprecated as well, in favour of the religiously neutral `BCE` favoured by science and academia. A future GEDCOM version will use `BCE` exclusively.

Developers that aim for maximum compatibility of GEDCOM 5.5.1 and 5.5.5 files should use `B.C.` in their GEDCOM 5.5.5 files for now, *and* make sure their GEDCOM 5.5.1 reader accepts `BCE`.

CASTE_NAME:= **{Size=1:90}**

A name assigned to a particular group that this person was associated with, such as a particular racial group, religious group, or a group with an inherited status.

CAUSE_OF_EVENT:= **{Size=1:90}**

Used in special cases to record the reasons which precipitated an event. Normally this will be used subordinate to a death event to show cause of death, such as might be listed on a death certificate.

CERTAINTY_ASSESSMENT:= **{Size=1:1}**

[0 | 1 | 2 | 3]
The QUAY tag's value conveys the submitter's quantitative evaluation of the credibility of a piece of information, based upon its supporting evidence. Some systems use this feature to rank multiple conflicting opinions for display of most likely information first. It is not intended to eliminate the receiver's need to evaluate the evidence for themselves.

0 = Unreliable evidence or estimated data

1 = Questionable reliability of evidence (interviews, census, oral genealogies, or potential for bias for example, an autobiography)
2 = Secondary evidence, data officially recorded sometime after event
3 = Direct and primary evidence used, or by dominance of the evidence

CERTAINTY_ASSESSMENT

This simple four-value certainty assessment is arguably too simple and too subjective for serious use. It is not used much.
Several modern genealogy applications use a multi-value quality assessment, which is recorded as a GEDCOM extension.

COPYRIGHT_GEDCOM_FILE:= {Size=1:248}

A copyright statement needed to protect the copyrights of the submitter of this GEDCOM file.

The copyright-ability of GEDCOM files is a tricky topic. A typical GEDCOM is a selection taken from a compilation of facts, interpretations, and notes, complete with mistakes. Fact aren't copyrightable, but compilations and original notes are, and your interpretations and mistakes are your own. Then again, a small selection does not necessarily enjoy the same copyright protection as the entire work. You need a lawyer if you really care about this topic.

COPYRIGHT_SOURCE_DATA:= {Size=1:248}

A copyright statement required by the owner of data from which this information was downloaded.
This value occurs in the GEDCOM header; usage of CONC or CONT records is not allowed.

Actual examples are hard to come by, as the HEAD.COPR is rarely used. This is a GEDCOM header produced by Ancestral File 4.19:

```
0 HEAD
1 SOUR ANSTFILE
2 VERS 4.19
2 NAME Ancestral File (R)
2 CORP The Church of Jesus Christ of Latter-day Saints

...
2 DATA Ancestral File
3 DATE 5 January 1998
3 COPR Copyright (c) 1987, June 1998
1 DEST PAF
1 DATE 6 FEB 2010
2 TIME 17:02:05
1 FILE GEDCOM4.ged
1 GEDC
2 VERS 5.5
2 FORM LINEAGE-LINKED
1 CHAR ANSEL
...
```

This example is part of an actual GEDCOM file created in 1998 which uses an old-style unstructured address, which were already deprecated but still legal in GEDCOM 5.5. To avoid giving a bad example, the old style address has been replaced with continuation dots.

COUNT_OF_CHILDREN:= {Size=1:3}

The known number of children of this individual from all relationships or, if subordinate to a

family group record, the reported number of children known to belong to this family group (couple), regardless of whether the associated children are represented in the corresponding structure. This is not necessarily equal to the number of children referenced in the family group record.

DATE:= **{Size=4:35}**

[
<DATE_CALENDAR>
|
<DATE_CALENDAR_ESCAPE> + space + <DATE_CALENDAR>
]

where:

space = U+0020, the space character

Date Calendar Escape Sequence

It is odd that FamilySearch's lineage-linked form uses a GEDCOM escape sequence to specify the calendar. The calendar could easily be expressed with say a DATE.TYPE record.
In fact, the GEDCOM 5.6 draft (2000 CE) gets rid of the escape sequence by doing just that; it introduces the optional DATE.CLNDR record.

Changes

FamilySearch kept changing its mind about the calendar escape sequence. FamilySearch GEDCOM 4.0 introduced the calendar escape sequences, and GEDCOM 5.0 still featured them. FamilySearch GEDCOM 5.3 features the calendar escape sequences, but GEDCOM 5.4 does not, and states that “The Lineage-Linked GEDCOM Form is restricted to Gregorian calendar forms.”. FamilySearch GEDCOM 5.5 and 5.5.1 feature the calendar escape sequences, and GEDCOM 5.6 replaces it with a subrecord.

DATE_APPROXIMATED:= **{Size=8:39}**

[
ABT + space + <DATE> |
CAL + space + <DATE> |
EST + space + <DATE>
]

where:

ABT = About, meaning the date is not exact.
CAL = Calculated mathematically, for example, from an event date and age.
EST = Estimated based on an algorithm using some other event date.
space = U+0020, the Space character

About Abt

The FamilySearch GEDCOM specification provides three different date modifiers for approximated dates, with three slightly different definitions. The implied demand to use different date modifiers for slightly different situations is not placed on genealogy software, but on users. In practice, users almost always use `ABT`. Few users even know that usage of `CAL` or `EST` is possible.

About Family Tree Maker Abt

The <DATE_RANGE> annotation Date Modifiers in Family Tree Maker provides a compatibility note about date modifiers in Family Tree Maker.

DATE_CALENDAR:= **{Size=4:35}**

[<DATE_GREG> | <DATE_JULN> | <DATE_HEBR> | <DATE_FREN>]
The selection is based on the <DATE_CALENDAR_ESCAPE> that precedes the <DATE_CALENDAR> value immediately to the left. If <DATE_CALENDAR_ESCAPE> doesn't appear at this point, then @#DGREGORIAN@ is assumed.
No future calendar types will use words (e.g., month names) from this list: `FROM`, `TO`, `BEF`, `AFT`, `BET`, `AND`, `ABT`, `EST`, `CAL`, or `INT`.
When only a day and month appears as a DATE value it is considered a date phrase and not a valid date form.

Date Escape	Syntax Selected
`@#DGREGORIAN@`	<DATE_GREG>
`@#DJULIAN@`	<DATE_JULN>
`@#DHEBREW@`	<DATE_HEBR>
`@#DFRENCH R@`	<DATE_FREN>
`@#DUNKNOWN@`	calendar not known

GEDCOM supports a limited number of calendars. For now, the escape sequence `@#DUNKNOWN@` should be used for dates in calendars not supported by GEDCOM.

Swedish Calendar

It is surprising that even GEDCOM version 5.5.1 still doesn't support the Swedish Calendar.
All it takes is adding the escape `@#DSWEDISH@` to the specification.
Conversion between the Swedish, Julian and Gregorian Calendars is trivially easy.

Sweden is the only country to not simply switch from the Julian to the Gregorian Calendar on a single day. Instead of simply skipping 11 calendar days as other countries had done, Sweden decided to gradually switch from the Julian to the Gregorian Calendar by omitting all the leap days in the period 1700 through 1740.
In 1700, Feb 29 was omitted from the Calendar. So, from that day forward, there was as Swedish Calendar that was one day behind the Julian Calendar, and ten days ahead of the Gregorian Calendar. The plan to omit the leap days from 1704 and 1708 wasn't executed: the difference with the Julian Calendar remained one day.
In 1711, Sweden decided to return to the Julian Calendar by adding an extra leap day to 1712, hence the existence of 30 Feb 1712. 30 Feb 1712 in the Swedish Calendar corresponds to 29 Feb 1712 in the Julian Calendar.
From 1 March 1712 forward, Sweden was using the Julian Calendar again, to eventually adopt the Gregorian Calendar in 1753.

Date without Year

The <DATE_CALENDAR> definition states that “When only a day and month appears as a DATE value it is considered a date phrase and not a valid date form.”.

Not allowing dates without a year is an unfortunate decision, and one that remains unmotivated.
The reason *is not* that a value like `23 Aug` is ambiguous, and could be understood to mean either the 23rd of August or August in the year 23; it *cannot* be interpreted as latter, because GEDCOM demands that years are at least 3 digits longs.

It is quite common for people to know birthdays without knowing the birth year, or remember a death date without the death year.
Knowing a date is so valuable in search for documents that could contain the year, that applications should support dates without a year. GEDCOM readers should accept such dates, but issue a warning that the year is missing.

DATE_CALENDAR_ESCAPE:= **{Size=4:15}**

[@#DHEBREW@ | @#DFRENCH R@ | @#DGREGORIAN@ | @#DJULIAN@ | @#DUNKNOWN@]
The date escape determines the date interpretation by signifying which <DATE_CALENDAR> to use. The default calendar is the Gregorian calendar.

Supported Calendars

The selection of supported calendars reveals a Western bias.
There are many calendars the GEDCOM specification does not support, including the Hijiri, Buddhist, Chinese, Japanese and Tibetan calendars.

Space in Calendar Escape

The French Calendar escape, `@#DFRENCH R@`, contains a space. This looks and feels like a mistake, but the GEDCOM grammar allows it, and it *is* what genealogy applications that support the French Revolutionary calendar use and expect.

Known Error

At least one developer assumed the space to be a mistake. Alsyd Parentèle, later MindScape Parentèle, now discontinued, exports `@#DFRENCHR@` (no space) instead of `@#DFRENCH R@`.
The GEDCOM lineage-linked form does not define a date escape `@#DFRENCHR@`. It is okay for a GEDCOM reader to issue a fatal error and abort upon encountering calendar escape `@#DFRENCHR@`, but it is suggested that GEDCOM readers issue a non-fatal error for each individual occurrence of `@#DFRENCHR@`, interpret it as `@#DFRENCH R@`, and continue processing.

The Gregorian Calendar escape is superfluous

The Gregorian Calendar is the default calendar, so the Gregorian Calendar escape (`@#DGREGORIAN@`) is technically superfluous.

GEDCOM Writer Best Practice

- do not use the Gregorian Calendar escape (`@#DGREGORIAN@`)

GEDCOM Reader Best Practice

- recognise and discard the Gregorian Calendar escape (`@#DGREGORIAN@`)

DATE_EXACT:= **{Size=10:11}**

<DAY> + space + <MONTH> + space + <YEAR>

where:

space = U+0020, the Space character

Date Validation

The FamilySearch GEDCOM specification says nothing about date validation. Genealogy applications should make sure that dates are valid, and that includes knowing the rules for leap years. Getting that right is a significant amount of work, so most developers will want to rely on existing date processing libraries.

References

Behold Blog: Out on Bad Date

DATE_FREN:= **{Size=4:35}**

[
<YEAR>
|
<MONTH_FREN> + space + <YEAR>
|
<DAY> + space + <MONTH_FREN> + space + <YEAR>
]

where:

space = U+0020, the Space character

See <MONTH_FREN>, page 94.

DATE_GREG:= **{Size=4:35}**

[
<YEAR> [+ space + <BEFORE_COMMON_ERA>]
|
<MONTH> + space + <YEAR>
|
<DAY> + space + <MONTH> + space + <YEAR>
|
<DAY> + space + <MONTH>
|
<MONTH> + space + <DUAL_STYLE_YEAR>
|

<DAY> + space + <MONTH> + space + <DUAL_STYLE_YEAR>
]

where:

space = U+0020, the Space character

Dual-style dates are explained in the <DUAL_STYLE_YEAR> definition.

DATE_HEBR:= **{Size=4:35}**

[
<YEAR>
|
<MONTH_HEBR> + space + <YEAR>
|
<DAY> + space + <MONTH_HEBR> + space + <YEAR>
]

where:

space = U+0020, the Space character

See <MONTH_HEBR, page 94.

Calendar Conversion

Genealogy application that show the Gregorian Date for a Hebrew date should take care when converting the date.

In the Hebrew calendar, days start at sunset. When faced with a date without time information, conversion between the Hebrew and Julian or Gregorian calendars should calculate the so-called *tabular date*, which is the corresponding date in the other calendar that has the same daylight period.

Anno Mundi

Previous versions of GEDCOM included an optional “B.C.” after a Hebrew year, which is most definitely a specification error.
That christian phrase is only used with proleptic Julian and Gregorian Calendars, it is never used with the Hebrew calendar.

A Hebrew date may be *preceded* by “AM” (without quotes), which is an abbreviation of *Anno Mundi*, Latin for “Year of the World”. Users who enter Hebrew dates (with or without the `@#HEBREW@` escape) directly into the date field are not unlikely to use this.
There is no proleptic Hebrew Calendar; the Hebrew Calendar is never used for dates before AM 1.

Best Practice

- GEDCOM writers should not write “AM”, as most GEDCOM readers won't recognise this
- GEDCOM readers should accept “AM”, as users may have entered this directly

DATE_JULN:= **{Size=4:35}**

[
<YEAR> [+ space + <BEFORE_COMMON_ERA>]

|
<MONTH> + space + <YEAR>
|
<DAY> + space + <MONTH> + space + <YEAR>
|
<MONTH> + space + <DUAL_STYLE_YEAR>
|
<DAY> + space + <MONTH> + space + <DUAL_STYLE_YEAR>]

where:

space = U+0020, the Space character

Dual-style dates are explained in the <DUAL_STYLE_YEAR> definition.

DATE_PERIOD:= **{Size=7:35}**

[
FROM + space + <DATE>
|
TO + space + <DATE>
|
FROM + space + <DATE> + space + TO + space + <DATE>
]

where:

FROM = Indicates the beginning of a happening or state.
TO = Indicates the ending of a happening or state.
space = U+0020, the space character

Examples:

FROM 1904 **TO** 1915

= The state of some attribute existed from 1904 to 1915 inclusive.

FROM 1904

= The state of the attribute began in 1904 but the end date is unknown.

TO 1915

= The state ended in 1915 but the begin date is unknown.

DATE_PERIOD TO means THROUGH

Note that DATE_PERIOD's TO is misnamed, and is used to mean THROUGH.

DATE_PHRASE:= **{Size=1:35}**

<TEXT>

Any statement offered in lieu of a date when the date is not recognisable to a date parser, but which gives information about when an event occurred.

A date phrase must be enclosed in matching parentheses. That demand is not immediately visible in the <DATE_PHRASE> definition, because it is made in the <DATE_VALUE> definition.

Date Phrase

The <DATE_PHRASE> is a bad idea because it enables *anything* in a DATE field. As a matter of principle, text that isn't a date should be in a NOTE record, not inside the DATE field itself.
It is strongly recommended that genealogy software tries to keeps DATE records clean, by working with the user to relegate “date phrases” to a NOTE.

The only text that isn't recognised as a date yet should still be in a date field is a valid date for a calendar that the application does not recognise or GEDCOM does not support. User can actually mark such text as a valid date, for example by including the escape `@FRENCH R@` if the application does not recognise French Revolutionary dates, or the escape `@DUNKNOWN@` for dates in any calendar not supported by GEDCOM.

References

GEDCOM date phrases

DATE_RANGE:= **{Size=8:35}**

[
BEF + space + <DATE>
| AFT + space + <DATE>
| BET + space + <DATE> + space + AND + space + <DATE>
]

where:

AFT = Event happened after the given date.
BEF = Event happened before the given date.
BET = Event happened sometime between date 1 AND date 2
space = U+0020, the Space character

For example, bet 1904 and 1915
indicates that the event state (perhaps a single day) existed somewhere between 1904 and 1915 inclusive.
The date range differs from the date period in that the date range is an estimate that an event happened on a single date somewhere in the date range specified.
The following are equivalent and interchangeable:

Short form	Long Form
1852	BET 1 JAN 1852 AND 31 DEC 1852
1852	BET 1 JAN 1852 AND DEC 1852
1852	BET JAN 1852 AND 31 DEC 1852
1852	BET JAN 1852 AND DEC 1852
JAN 1920	BET 1 JAN 1920 AND 31 JAN 1920

Date Modifiers in Family Tree Maker

The date modifiers `ABT`, `BEF` & `AFT` are three-letter abbreviations. Several old versions of Family Tree Maker use the illegal four-letter abbreviations `ABT.`, `BEF.` & `AFT.` instead; Family Tree Maker adds a fourth character, a full stop, to the abbreviation while it should not.

A GEDCOM 5.5.5 reader *must* support the official date modifiers and *must not*

support any others.
GEDCOM readers for older GEDCOM versions need not accept these illegal date modifiers, but it is trivial to support. Even a GEDCOM reader that supports it should still issue a non-fatal error for each occurrence.

Date Range in FamilySearch PAF

The ability to specify a <DATE_RANGE> is an important feature. Oddly, FamilySearch PAF does not support it, but pops up a message box stating "The date is not standard" when you use it. You can still use date ranges; PAF will (erroneously) treat the <DATE_RANGE> as a <DATE_PHRASE>, and export it exactly the way you entered it (without the enclosing parentheses mandatory for date phrases), so genealogy applications that do support date ranges will import it just fine.
Incidentally, PAF *does* support <DATE_PERIOD>.

DATE_VALUE:= **{Size=1:35}**

[
<DATE>
|
<DATE_PERIOD>
|
<DATE_RANGE>
|
<DATE_APPROXIMATED>
|
(<DATE_PHRASE>)
|
INT + space + <DATE> + space + (<DATE_PHRASE>)
]

The <DATE_VALUE> represents the date of an activity, attribute, or event where:

INT = Interpreted from knowledge about the associated date phrase included in parentheses.
space = U+0020, the Space character

An acceptable alternative to the date phrase choice is to use one of the other choices such as <DATE_APPROXIMATED> choice as the DATE line value and then include the date phrase value as a NOTE value subordinate to the DATE line tag.

The date value can take on the date form of just a date, an approximated date, between a date and another date, and from one date to another date. The preferred form of showing date imprecision, is to show, for example, `MAY 1890` rather than `ABT 12 MAY 1890`. This is because limits have not been assigned to the precision of the prefixes such as `ABT` or `EST`.

Approximate Exact Date

The <DATE_VALUE> definition does allow `ABT` in front of an exact date; the value `ABT 12 May 1890` is legal, but applications should warn the user that `ABT` in front of an exact date specifies an approximate exact date, which seems self-contradictory. Practically all cases of `ABT` in front of an exact date should be either `BEF` (before) or `AFT` (after) in front of that date.

Date Phrases in Parentheses

Note the parentheses around <DATE_PHRASE>; the <DATE_VALUE> definition demands that all date phrases be enclosed in parentheses.

Date Phrases in PAF

FamilySearch PAF 5 includes <DATE_PHRASE> support, but PAF exports date phrases to GEDCOM exactly the way users enter them, *without* the mandatory enclosing parentheses. This is wrong.

FamilySearch GEDCOM 5.5.1 PAF DATE record abusage

FamilySearch's own PAF does not respect the <DATE_VALUE> definition, but abuses date fields in ways not intended or allowed by the FamilySearch GEDCOM specification. PAF includes phrases such as `NOT MARRIED`, `CHILD`, `STILLBORN` and `INFANT` in date fields (without the parentheses mandatory for date phrases).

PAF development was abandoned in 2012. There is no such thing as a PAF GEDCOM 5.5.5 file. A GEDCOM 5.5.5 reader *need not* and *must not* support these PAF forms.

Best Practice

GEDCOM Writer Best Practice

- Always use parentheses around date phrases as demanded by the GEDCOM specification

GEDCOM Reader Best Practice

- Make sure you recognise dates in all the supported calendars
- Treat every date value you do not recognise as a date phrase
- Issue an error for date phrase not enclosed in parentheses

References

Behold blog: Out on a Bad Date

DAY:= **{Size=1:2}**

dd
Day of the month, where dd is a numeric value within the valid range of the days for the associated calendar month.
This value *must not* start with a leading zero.

Date Validation

The description states that the day value must be legal; i.e. the value must be within the valid range of dates for the month.
This demand implies that *genealogy software is expected to validate dates* upon both user input and GEDCOM import.
When validating dates, applications should make sure that the day not only fits the

month, but - for leap days - the year and calendar as well.

30 Feb 1712

The date 30 Feb 1712 was a real date in Sweden. There are records with that date. 30 Feb 1712 on the Swedish Calendar corresponds with 29 Feb 1712 on the Julian Calendar. 30 Feb 1712 isn't a date on either the Julian or Gregorian calendar, but only on the Swedish Calendar, which even GEDCOM version 5.5.1 still fails to support.

DESCRIPTIVE_TITLE:= **{Size=1:248}**

The title of a work, record, item, or object.

DIGIT:= **{Size=1:1}**

A single digit (0-9).

DUAL_STYLE_YEAR:= **{Size=3:7}**

<YEAR> + slash + <DIGIT> + <DIGIT>

where:

slash = U+002F, the Slash character

The three-character part after the year (slash + <DIGIT> + <DIGIT>) is the alternate year indicator.
The alternate year indicator does *not* indicate that the date is uncertain. It creates *dual style date*, which indicates that the year is either the one or the other year, depending on the calendar style. The full year is the Old Style year, the two digits indicate the New Style year.

Dual-Style Dates

The alternate year indicator always has two digits, never one or three, and always indicates the subsequent year. The alternate date indicator must consist of the last two digits of the next year; anything else is a syntax error and *not a date*.

Old Style and New Style

There are two calendar styles, the Old Style (OS) calendar and the New Style (NS) calendar. The Old Style calendar starts the year on the 25th of March while the New Style starts the year on the 1st of January.

Calendar Act

Before 1752, the English used both what was then known as the Historical Year, starting on January 1, and the Civil Year or Legal Year, starting on March 25. That is why Dual-style dates are common in pre-1752 English parish registers.
The Calendar Act of 1750 ended the practice of using both calendar styles by changing the Legal Year to start on January 1, just like the Historical Year.

Many European countries switched from the Julian to the Gregorian Calendar in 1582, England continued to use the Julian Calendar through 1752. The Calendar Act of 1750 enacted two changes; it changed the calendar date on which the Legal Year begins, and it changed the calendar from the Julian Calendar to the Gregorian Calendar. The Legal Year 1751 ran from 25 March 1751 till 1 January 1752. Wednesday 2 September 1752 JC was followed by Thursday 14 Sep 1752 GC.

Dual Style resolves Ambiguity

The simultaneous use of Old Style and New Style before 1752 created ambiguity; the 21st of February might be in the year 1750 according to the Old Style calendar, but in the year 1751 according to the New Style calendar. I could write “21 Feb 1751” hoping that you assume it is New Style, but I might assume wrongly, and if you believe it to be an Old Style date, you might misinterpret it as 21 Feb 1752 New Style. Writing it as“21 Feb 1750/51” disambiguates the date, by providing the years for both styles: it explicitly states that it is 21 Feb 1750 Old Style and 21 Feb 1751 New Style, no confusion possible.

The Old Style year is listed first, the New Style year is listed second. The second year is always the next year, is always indicated by exactly two digits, the last two digits of that year. So “21 Feb 1750/52” is illegal, “21 Feb 1759/60” involves the years 1759 and 1760, while “21 Feb 759/0” is illegal. Even “21 Feb 1759/1760”, arguably an easier to understand more straightforward syntax, is illegal.

Usage

Dual styling may be needed for dates in January, February and most of March (up to, but not including March 25). For the rest March, and all of April through December, the Old Style calendar and the New Style calendar agree on the year, so those dates must not be dual-styled. Using dual-style for a date that isn't ambiguous does not create ambiguity, it creates *not a date*.

Historic Names

Historically, many authors and publications have referred to dual style dates as double dates or dual dating. Those ill-considered names create confusion by suggesting that there are two different dates, while that is not the case at all. There is a single date, in two different calendar styles, hence *dual style date*.

Dual Calendaring

There is also a dual-calendar practice used because of the change from the Julian Calendar (JC) to the Gregorian Calendar (GC, usually indicated with CE). For example, “2/ 12 May 1608” is 2 May 1608 JC and 12 May 1608 CE.
This dual-calendaring practice is not supported by GEDCOM; GEDCOM demands that you pick a calendar.
Within GEDCOM, the Gregorian Calendar is the default calendar, but several other calendars, including the Julian Calendar, can be specified through the <DATE_CALENDAR_ESCAPE>.

Restrictions

Genealogy applications must ensure correct usage of dual-style dates by enforcing appropriate restrictions.

- Dual style dates may only be used with the Julian and Gregorian Calendar
- The use of two years is indicated by a slash (never a hyphen or any other character)
- The slash must always be followed by two digits, never more or less
- The year indicated by those digits must be the next year
- Dual style may only be used for otherwise ambiguous dates, so only for Jan 1 through March 24
- Dual style may only be used for the year 1923 and earlier

References

GEDCOM Dual-Style Dates

ENTRY_RECORDING_DATE:= **{Size=1:90}**

<DATE_VALUE>
The date that this event data was entered into the original source document.

EVENT_ATTRIBUTE_TYPE:= **{Size=1:15}**

[<EVENT_TYPE_INDIVIDUAL> |
<EVENT_TYPE_FAMILY> |
<ATTRIBUTE_TYPE>]

A code that classifies the principal event or happening that caused the source record entry to be created. If the event or attribute doesn't translate to one of these tag codes, then a user supplied value is expected and will be generally classified in the category of other.

EVENT_DESCRIPTOR:= **{Size=1:90}**

Text describing a particular event pertaining to the individual or family. This event value is usually assigned to the EVEN tag. The classification as to the difference between this specific event and other occurrences of the EVEN (event) tag is indicated by the use of a subordinate TYPE tag selected from the <EVENT_DETAIL> structure. For example:

1 EVEN Appointed Zoning Committee Chairperson
2 TYPE Civic Appointments
2 DATE FROM JAN 1952 TO JAN 1956
2 PLAC Cove, Cache, Utah, United States of America
2 AGNC Cove City Redevelopment

EVENT_OR_FACT_CLASSIFICATION:= **{Size=1:90}**

A descriptive word or phrase used to further classify the parent event or attribute tag. This should be used whenever either of the generic **EVEN** or **FACT** tags are used. The value of this primitive is responsible for classifying the generic event or fact being cited. For example, if the attribute being defined was one of the persons skills, such as woodworking, the **FACT** tag would have the value of `Woodworking', followed by a subordinate TYPE tag with the value `Skills'.

1 FACT Woodworking
2 TYPE Skills

This groups the fact into a generic skills attribute, and in particular this entry records the fact that this individual possessed the skill of woodworking. Using the subordinate TYPE tag classification method with any of the other defined event tags provides a further classification of the parent tag but does not change the basic meaning of the parent tag. For example, a **MARR** tag could be subordinated with a **TYPE** tag with an <EVENT_OR_FACT_CLASSIFICATION> value of `Common Law.'

1 MARR
2 TYPE Common Law

This classifies the entry as a common law marriage but the event is still a marriage event. Other descriptor values might include, for example,`stillborn' as a qualifier to BIRT (birth) or `Tribal Custom' as a qualifier to MARR (relationship).

The <EVENT_OR_FACT_CLASSIFICATION> is an open-ended feature.
There is no list of pre-defined, accepted or suggested descriptor values.

TYPE tag Correction

The FamilySearch GEDCOM 5.5.1 description for <EVENT_OR_FACT_CLASSIFICATION> references <EVENT_DESCRIPTOR> instead of <EVENT_OR_FACT_CLASSIFICATION> as intended.

This is an editing error, made upon the introduction of FACT and FACT.TYPE in GEDCOM 5.5.1. This specification error has been highlighted in the *Annotated Edition* and corrected in GEDCOM 5.5.5.

This error was first noted by Keith Riggle and mentioned in several software reviews. It was documented by Tim Forsythe in his 22 Dec 2015 GigaTrees blog post *A New GEDCOM 5.5.1 Wrinkle*, but that blog post is no longer available. It is now documented in Keith Riggle's own blog post *The Event Structure in GEDCOM Files*.

References

GenealogyTools: The Event Structure in GEDCOM Files

EVENT_TYPE_CITED_FROM:= **{Size=1:15}**

[<EVENT_ATTRIBUTE_TYPE>]

A code that indicates the type of event which was responsible for the source entry being recorded. For example, if the entry was created to record a birth of a child, then the type would be BIRT regardless of the assertions made from that record, such as the mother's name or mother's birth date. This will allow a prioritised best view choice and a determination of the certainty associated with the source used in asserting the cited fact.

EVENT_TYPE_FAMILY:= **{Size=3:4}**

[ANUL | CENS | DIV | DIVF | ENGA | MARR |
MARB | MARC | MARL | MARS | EVEN]

A code used to indicate the type of family event. The definition is the same as the corresponding event tag defined in Appendix A. (See Appendix A, page 121).

EVENT_TYPE_INDIVIDUAL:= **{Size=3:4}**

[ADOP | BIRT | BAPM | BARM | BASM |
BURI | CENS | CHR | CHRA |
CONF | CREM | DEAT | EMIG | FCOM |
GRAD | IMMI | NATU |
RETI | PROB | WILL | EVEN]

A code used to indicate the type of individual event. The definition is the same as the corresponding event tag defined in Appendix A. (See Appendix A, page 121).

EVENTS_RECORDED:= **{Size=1:90}**

[<EVENT_ATTRIBUTE_TYPE> |
<EVENTS_RECORDED>, <EVENT_ATTRIBUTE_TYPE>]

An enumeration of the different kinds of events that were recorded in a particular source. Each enumeration is separated by a comma. Such as a parish register of births, deaths, and marriages would be BIRT, DEAT, MARR.

FILE_CREATION_DATE:= **{Size=10:11}**

<DATE_EXACT>
The date that this GEDCOM file was created.

GEDCOM_CONTENT_DESCRIPTION:= **{Size=1:248}**

A note that a user enters to describe the contents of the lineage-linked file in terms of "ancestors or descendants of" so that the person receiving the data knows what genealogical information the file contains.
This value occurs in the GEDCOM header; usage of CONC or CONT records is not allowed.

GEDCOM_FILE_NAME:= **{Size=5:248}**

The name of the GEDCOM file. A GEDCOM file name should use the format *basename.ext*, and use the file extension `.GED` (or `.ged`). GEDCOM 5.5.5 increases the maximum file name length from 90 to 248 code units.

The <GEDCOM_FILE_NAME> is specified as the HEAD.FILE line value. The HEAD.FILE line value should match the name of the GEDCOM file itself, but when a GEDCOM file is renamed by a user, the HEAD.FILE line value does not change with the file name.

<GEDCOM_FILE_NAME> should be just the regular file name (base name plus extension), *without* a file path. The file name should *not* be enclosed in parentheses.

Whether file names are case-sensitive is platform-dependant. All GEDCOM reader and writer file handling code should be case-preserving.

GEDCOM Writer Checklist

- The line value must be the name of the GEDCOM file itself.
- Use only the regular filename (basename plus extension)
- *Do not* include a file path
- *Do* include the file extension.
- Use the file extension `.GED` or `.ged`
- *Do not* use parentheses around the name.
- File handling code must be case-preserving.

ID_NUMBER:= **{Size=1:30}**

A third-party number assigned to an individual.
This IDNO line value must be used for all third party numbers. The IDNO record has a TYPE subrecord to identify what kind of number is being stored.

For example:

1 IDNO 43-456-1899
2 TYPE Canadian Health Registration

The IDNO record is not restricted to national numbers. The IDNO record should be used for all third-party numbers, and the REFN record for all user-defined numbers.

LANGUAGE_ID:= **{Size=1:15}**

A list of valid language identification codes.

The following Latin language codes may be used by all systems (including code page-based systems):

[Afrikaans | Albanian | Anglo-Saxon | Catalan | Catalan_Spn | Czech | Danish | Dutch |

English | Esperanto | Estonian | Faroese | Finnish | French | German | Hawaiian | Hungarian | Icelandic | Indonesian | Italian | Latvian | Lithuanian | Navaho | Norwegian | Polish | Portuguese | Romanian | Serbo_Croa | Slovak | Slovene | Spanish | Swedish | Turkish | Wendic]

Additional languages supported by Unicode-based systems:

[Amharic | Arabic | Armenian | Assamese | Belorusian | Bengali | Braj | Bulgarian | Burmese | Cantonese | Church-Slavic | Dogri | Georgian | Greek | Gujarati | Hebrew | Hindi | Japanese | Kannada | Khmer | Konkani | Korean | Lahnda | Lao | Macedonian | Maithili | Malayalam | Mandarin | Manipuri | Marathi | Mewari | Nepali | Oriya | Pahari | Pali | Panjabi | Persian | Prakrit | Pusto | Rajasthani | Russian | Sanskrit | Serb | Tagalog | Tamil | Telugu | Thai | Tibetan | Ukrainian | Urdu | Vietnamese | Yiddish]

LANGUAGE_OF_TEXT:= **{Size=1:15}**

[<LANGUAGE_ID>]
The human language in which the data in the GEDCOM file is normally read or written. It is used primarily by programs to select language-specific sorting sequences and phonetic name matching algorithms.

MONTH:= **{Size=3}**

[JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC]

where:

JAN = January
FEB = February
MAR = March
APR = April
MAY = May
JUN = June
JUL = July
AUG = August
SEP = September
OCT = October
NOV = November
DEC = December

MONTH_FREN:= **{Size=4}**

[VEND | BRUM | FRIM | NIVO | PLUV | VENT | GERM | FLOR | PRAI | MESS | THER | FRUC | COMP]

where:

VEND = VENDEMIAIRE
BRUM = BRUMAIRE
FRIM = FRIMAIRE
NIVO = NIVOSE
PLUV = PLUVIOSE
VENT = VENTOSE
GERM = GERMINAL
FLOR = FLOREAL
PRAI = PRAIRIAL
MESS = MESSIDOR
THER = THERMIDOR
FRUC = FRUCTIDOR
COMP = JOUR_COMPLEMENTAIRS

MONTH_HEBR:= **{Size=3}**

[TSH | CSH | KSL | TVT | SHV | ADR | ADS | NSN | IYR | SVN | TMZ | AAV | ELL]

where:

TSH = Tishri
CSH = Cheshvan
KSL = Kislev
TVT = Tevet
SHV = Shevat
ADR = Adar
ADS = Adar Sheni
NSN = Nisan
IYR = Iyar
SVN = Sivan
TMZ = Tammuz
AAV = Av
ELL = Elul

MULTIMEDIA_FILE_REFERENCE:= {Size=1:259}

A complete local or remote file reference to the auxiliary data to be linked to the GEDCOM context. Remote reference would include a network address where the multimedia data may be obtained.

The <MULTIMEDIA_FILE_REFERENCE> definition is for multimedia files only.
The GEDCOM specification has a separate <GEDCOM_FILE_NAME> definition for the HEAD.FILE line value.

Whether file names are case-sensitive is platform-dependant. Make sure your file handling code is case-preserving.

The maximum path length specified by GEDCOM 5.5.1 was only 30 code units, which isn't a realistic maximum length for a full path.
The *Annotated Edition* corrected the maximum to be 259 code units. GEDCOM 5.5.5 makes that the official maximum length.

MULTIMEDIA_FORMAT:= {Size=3:4}

[AAC | AVI | BMP | ePub | FLAC | GIF | JPEG | JPG | MKV | mobi | MP3 | PCX | PDF | PNG | TIFF | TIF | WAV]

where:

AAC = Advanced Audio Codec
AVI = Audio Video Interleaved (Windows)
BMP = BitMaP (Windows)
ePUB = Electronic Publication (ebook)
FLAC = Free Lossless Audio Codec
GIF = Graphics Interchange Format (CompuServe)
JPEG, **JPG** = Joint Photographic Experts Group
MKV = Matroska Video Container
mobi = MobiPocket (ebook)
MP3 = MPEG-2 Audio Layer III
PCX = Personal Computer eXchange (PaintBrush)
PDF = Portable Document Format
PNG = Portable Network Graphics
TIFF, **TIF** = Tagged Image File Format
WAV = WAVeform audio file format (Windows)

Indicates the format of the multimedia data associated with the specific GEDCOM context. This allows processors to determine whether they can process the data object. Any linked files should contain the data required, in the indicated format, to process the file data. GEDCOM readers must accept any casing for the enumerated formats, but GEDCOM writers should use the casing preferred in text written for humans.

GEDCOM 5.5.5 expands the list of allowed multimedia formats. The fact that the enumeration of allowed formats is supposed to be exhaustive remains a fundamental issue.

Four-Letter `JPEG` & `TIFF` versus Three-Letter `JPG` & `TIF`

The GEDCOM 5.5 specification includes the four-letter values `JPEG` and `TIFF`, while the GEDCOM 5.5.1 specification includes the three-letter values `JPG` and `TIF`, probably because some editor thought they should allow three-letters values only. That edit should not have been made, because now the two three-letter values are invalid in GEDCOM 5.5 while the four-letter values are invalid in GEDCOM 5.5.1.
Of course, in practice, most GEDCOM 5.5.1 reader accept the four-letter value without so much as a warning, but many GEDCOM 5.5 readers do not accept the three-letter values.

GEDCOM 5.5.5 uses the four-letter abbreviations, not only because those are the original abbreviations, but also because these are ones already supported by *both* GEDCOM 5.5 and GEDCOM 5.5.1 readers.
A GEDCOM 5.5.5 writer must use `JPEG` & `TIFF` exclusively. Systems must continue to support `JPG` & `TIF` when reading GEDCOM 5.5.1 files.

NAME_OF_BUSINESS:= **{Size=1:90}**

Name of the business, corporation, or person that produced or commissioned the product.

NAME_OF_PRODUCT:= **{Size=1:90}**

The name of the software product that produced this GEDCOM file.

NAME_OF_REPOSITORY:= **{Size=1:90}**

The official name of the archive in which the stated source material is stored.

NAME_OF_SOURCE_DATA:= **{Size=1:90}**

The name of the electronic data source that was used to obtain the data in this GEDCOM file. For example, the data may have been obtained from a CD-ROM disc that was named “U.S. 1880 CENSUS CD-ROM vol. 13”.

NAME_PERSONAL:= **{Size=1:120}**

[
<NAME_TEXT>
|
/<NAME_TEXT>/
|
<NAME_TEXT> + space + /<NAME_TEXT>/
|
/<NAME_TEXT>/ + space + <NAME_TEXT>
|
<NAME_TEXT> + space + /<NAME_TEXT>/ + space + <NAME_TEXT>
]

where:

space = U+0020, the Space character

The surname of an individual, if known, is enclosed between two slash (/) characters. The order of the name parts should be the order that the person would, by custom of their culture, have used when giving it to a recorder. Early versions of Personal Ancestral File ® and other products did not use the trailing slash when the surname was the last element of the name. If part of a name is illegible, that part is indicated by an ellipsis (...). Capitalize the name of a person or place in the conventional manner — capitalize the first letter of each part and lowercase the other letters, unless conventional usage is otherwise. For example: McMurray.

Examples

William Lee /Parry/	given name William Lee, surname Parry
William Lee	given name only or surname not known
/Parry/	surname only
William Lee /Mac Parry/	both Mac and Parry are part of the surname Mac Parry
William /Parry/ Boatsman	surname Parry embedded in the name string
William Lee /Pa.../	surname partly unknown (unreadable)

Name Guidelines

The FamilySearch GEDCOM specification for <NAME_PERSONAL> includes a few quick guidelines on how to record names. The key guideline provided here is to capitalise names in the conventional manner (so *not* in ALL-CAPS).
The article *Genealogy Name Basics* provides more *do's and don'ts*.

References

- Genealogy Name Basics
- Five Freaky Features Your Genealogy Software should *not* have

NAME_PHONETIC:= **{Size=1:120}**

The phonetic variation of the name is written in the same form as the name used in the superior <NAME_PERSONAL> primitive, but phonetically written using the method indicated by the subordinate <PHONETISATION_METHOD> value, For example if hiragana was used to provide a phonetic transcription of a name written in kanji. then the <PHONETISATION_METHOD> value would indicate ‘kana’. See <PHONETISATION_METHOD>, page 100 .

NAME_PIECE:= **{Size=1:90}**

The piece of the name pertaining to the name part of interest. The surname part, the given name part, the name prefix part, or the name suffix part.

NAME_PIECE_GIVEN:= **{Size=1:120}**

[<NAME_PIECE> | <NAME_PIECE_GIVEN>, <NAME_PIECE>] Given name or earned name. Different given names are separated by a comma.

NAME_PIECE_NICKNAME:= **{Size=1:30}**

[<NAME_PIECE> | <NAME_PIECE_NICKNAME>, <NAME_PIECE>] A descriptive or familiar name used in connection with one's proper name.

NAME_PIECE_PREFIX:= **{Size=1:30}**

[<NAME_PIECE> | <NAME_PIECE_PREFIX>, <NAME_PIECE>]
Non indexing name piece that appears preceding the given name and surname parts. Different name prefix parts are separated by a comma.
For example:
Lt. Cmndr. Joseph /Allen/ jr.
In this example **Lt. Cmndr.** is considered as the name prefix portion.

Nobility Titles

The <NAME_PIECE_PREFIX> should not be used for recording nobility titles. Nobility titles must be recorded as the <NOBILITY_TYPE_TITLE> line value of an INDI.TITL record.

NAME_PIECE_SUFFIX:= {Size=1:30}

[<NAME_PIECE> | <NAME_PIECE_SUFFIX>, <NAME_PIECE>]
Non-indexing name piece that appears after the given name and surname parts. Different name suffix parts are separated by a comma.
For example:
Lt. Cmndr. Joseph /Allen/ **jr.**
In this example **jr.** is considered as the name suffix portion.

NAME_PIECE_SURNAME:= {Size=1:120}

[<NAME_PIECE> | <NAME_PIECE_SURNAME>, <NAME_PIECE>] Surname or family name. Different surnames are separated by a comma.

NAME_PIECE_SURNAME_PREFIX:= {Size=1:30}

[<NAME_PIECE>]

Surname prefix or article used in a family name.

A surname prefix that consists of multiple parts is written *as is*, and not modified in any way. Thus, the surname prefix for the surname “de la Cruz” is “de la”.

FamilySearch's instruction, present in GEDCOM 5.5 and 5.5.1, to comma-separate parts of the surname prefix, and write “de, la” instead of “de la” *must be ignored*, even in GEDCOM 5.5 and 5.5.1 files.
That erroneous instruction remains unmotivated, and no genealogy application is known to follow it. Modifying a surname prefix by inserting one or more commas is sure to create those superfluous commas in your reports.

NAME_ROMANISED:= {Size=1:120}

The romanised name is written in the same form prescribed for the name used in the superior <NAME_PERSONAL> context. The method used to romanise the name is indicated by the line value of the subordinate <ROMANISATION_METHOD>. For example if romaji was used to provide a transliteration of a name written in kanji, then the <ROMANISATION_METHOD> subordinate to the ROMN tag would indicate `romaji`.
See <ROMANISATION_METHOD>, page 105.

NAME_TEXT:= {Size=1:120}

<TEXT> excluding commas, numbers, special characters not considered diacritics.

NAME_TYPE:= {Size=5:30}

[aka | birth | immigrant | maiden | married | <user defined>]

Indicates the name type, for example the name issued or assumed as an immigrant.

aka = also known as, alias, etc.
birth = name given on birth certificate.
immigrant = name assumed at the time of immigration.
maiden = maiden name, name before first marriage.
married = name was person's previous married name.

user defined = other text name that defines the name type.

NATIONAL_OR_TRIBAL_ORIGIN:= {Size=1:120}

The person's division of national origin or other folk, house, kindred, lineage, or tribal interest. Examples: Irish, Swede, Egyptian Coptic, Sioux Dakota Rosebud, Apache Chiricawa, Navajo Bitter Water, Eastern Cherokee Taliwa Wolf, and so forth.

NOBILITY_TYPE_TITLE:= {Size=1:120}

The title given to or used by a person, especially of royalty or other noble class within a locality.

NULL:= {Size=0:0}

A convention that does *not* indicate the line value `NULL`, nor the Null character (U+0000), but the complete absence of a line value.

NUMBER:= {Size=3:4}

[<DIGIT> | <NUMBER>+<DIGIT>]

> **<NUMBER> Size**
>
> <NUMBER> definition only occurs in the definition of <YEAR_GREG>. As a year is at least three digits long (see <YEAR_GREG> definition), at most 4 digits long; GEDCOM will surely have been replaced before 10.000 CE.

NUMBER_OF_RELATIONSHIPS:= {Size=1:3}

The number of different relationships (family groups) that this person was known to have been a member of as a partner, regardless of whether the associated relationships are present in the GEDCOM file.

OCCUPATION:= {Size=1:90}

The kind of activity that an individual does for a job, profession, or principal activity.

PEDIGREE_LINKAGE_TYPE:= {Size=5:7}

[adopted | birth | foster]

where:

adopted = indicates adoptive parents.
birth = indicates official parents (birth parents).
foster = indicates child was included in a foster or guardian family.

A code used to indicate the child to family relationship for pedigree navigation purposes. When <PEDIGREE_LINKAGE_TYPE> is absent, official parentage (`birth`) is assumed.

GEDCOM 5.5.5 is essentially GEDCOM 5.5.1 from 1999. GEDCOM 5.5.1 predates both affordable consumer DNA testing and the Genealogy Framework. That explains why GEDCOM 5.5.5 supports official genealogy (birth parents are the parents on the birth certificate), but does not support biological genealogy (parents confirmed by DNA tests) yet.

PHONE_NUMBER:= {Size=1:25}

A phone number.

PHONETISATION_METHOD:= **{Size=5:30}**

[<user defined> | hangul | kana]

The phonetisation method used for creating the phonetic text.

<user defined> = record method used to arrive at the phonetic variation of the name.
Hangul = Phonetic method for transcribing Korean glyphs.
kana = Hiragana and/or Katakana characters were used in transcribing the kanji character used by Japanese

PHYSICAL_DESCRIPTION:= **{Size=1:4095}**

An unstructured list of the attributes that describe the physical characteristics of a person, place, or object.
Commas separate each attribute.
Example:

1 DSCR Hair Brown, Eyes Brown, Height 5 ft 8 in
2 DATE 23 JUL 1935

PLACE_LATITUDE:= **{Size=2:10}**

The value specifying the latitudinal coordinate of the place name. The latitude coordinate is the direction North or South from the equator in degrees and fraction of degrees carried out to give the desired accuracy. For example: 18 degrees, 9 minutes, and 3.4 seconds North would be formatted as N18.150944. Minutes and seconds are converted by dividing the minutes value by 60 and the seconds value by 3600 and adding the results together. This sum becomes the fractional part of the degree's value.

The shortest possible value is "N0", the longest possible value, with 6 decimals precision is "N89.123456".

PLACE_LONGITUDE:= **{Size=2:11}**

The value specifying the longitudinal coordinate of the place name. The longitude coordinate is Degrees and fraction of degrees east or west of the zero or base meridian coordinate. For example: 168 degrees, 9 minutes, and 3.4 seconds East would be formatted as E168.150944.

The shortest possible value is "E0", the longest possible value, with 6 decimals precision is "E179.123456".

Latitude and Longitude Standard

The FamilySearch GEDCOM specification provides a brief specification of how to record latitude and longitude in the <PLACE_LATITUDE> and <PLACE_LONGITUDE> definitions.

Standard formats for geographical coordinates are defined ISO 6709 *Standard representation of geographic point location by coordinates*.
This format always uses a full stop (.), never a comma (,), as a decimal stop.
Use of the characters N and S for North and South, and E and W for East and West as sign characters is defined in Annex H.

Several applications use the plus (+) and minus (-) sign as sign characters.
GEDCOM readers should accept both styles.

PLACE_NAME:= **{Size=1:120}**

[

<PLACE_TEXT> |
<PLACE_TEXT>, + space + <PLACE_NAME>
]

where:

space = U+0020, the Space character

The jurisdictional name of the place where the event took place. Jurisdictions are separated by a comma and space combination. For example: "Cove, Cache, Utah, United States of America".
No part of the place name may be replaced by an abbreviation. Place names are not terminated by a full stop or anything else.

PLACE_PHONETIC:= **{Size=1:120}**

The phonetic variation of the place name is written in the same form as was the place name used in the superior <PLACE_NAME> primitive, but phonetically written using the method indicated by the subordinate <PHONETISATION_METHOD> value. For example if hiragana was used to provide a phonetic transcription of a name written in kanji, then the <PHONETISATION_METHOD> value would indicate kana. (See PHONETISATION_METHOD, page 100.)

PLACE_ROMANISED:= **{Size=1:120}**

The romanised transcription of the place name is written in the same form prescribed for the place name used in the superior <PLACE_NAME> context. The method used to romanise the name is indicated by the line value of the subordinate <ROMANISATION_METHOD>, for example if romaji was used to provide a transcription of a place name written in kanji, then the <ROMANISATION_METHOD> subordinate to the ROMN tag would indicate ‘romaji’ (See ROMANISATION_METHOD.), page 105

PLACE_TEXT:= **{Size=1:120}**

<TEXT> excluding any commas.

A fully specified place name exists of several parts, from place name up to country, separated by comma & space combinations, <PLACE_TEXT> is one such place name part. As place name parts are separated by commas, including any commas within a place name part would create confusion, so they have to be left out.

Example

An example of a place name that includes a comma is “De Mijl, Krabbe en Nadort” (without quotes), which existed in the province of Noord-Brabant in the Netherlands. That place should be specified as “De Mijl Krabbe en Nadort, Noord-Brabant, Netherlands” (without quotes).

Violation of Place Name Guidelines

The demand to leave out commas violates basic place name guidelines. *The* basic guideline is to enter place name parts *as-is*, and that means that you should always use full names, never abbreviations, and always use just the actual name, without anything added to it, and without leaving anything out.

The GEDCOM specification demands one exception to this rule: leave commas out of place names. The GEDCOM specification makes that demand because GEDCOM cannot support place names that include commas correctly.
The FamilySearch GEDCOM creators probably thought such place names do not exist, and later discovered that such names do exist. You should include such place

names without their comma, just as the FamilySearch GEDCOM specification demands.

References

Place Name Standardisation Basics

POSSESSIONS:= **{Size=1:248}**

A list of possessions (real estate or other property) belonging to this individual.

PRODUCT_VERSION_NUMBER:= **{Size=3:15}**

MMM + dot + mmm [+ dot + rrr [+ dot + bbb]]

where:

MMM = 1 through 3 digits; the major version number
mmm = 1 through 3 digits; the minor version number
rrr = 1 through 3 digits; the revision number
bbb = 1 through 3 digits; the build number
dot = U+002E, the Full Stop character

The version of the product that created the GEDCOM file.
The product version number is controlled by the developers of the product, but *must* be in this format, and *should* comply with common software industry version number practices briefly mentioned here.

The product version number *must* be provided through <PRODUCT_VERSION_NUMBER> and should not occur anywhere else. Specifically, the version number *must not* be used as part of the product name, but it is allowed to use a *marketing version* in the product name; e.g. “Family Tree Maker 2017” is a marketing version of the Family Tree Maker product. System version and build information is conveyed through HEAD.SOUR.VER and through HEAD.SOUR.VERS only.

Four Values

A product version consists of at least two and most four dot-separated values in *major.minor.revision.build* format. The four parts of the *major.minor.revision.build* format are:

The four parts of the *major.minor.revision.build* format are:

major = the major version number; starts at 0 (zero).
minor = the minor version number; starts at 0 for each new *major* version.
revision = the revision number; starts at 0 for each new *major.minor* version.
build = the build number for *major.minor.revision*.

Dot-Separated Values

The full stops used in version numbers are known as dots, but pronounced as “point”. A version number never begins or ends with a dot, it always begins and ends with a digit.
Values used must appear in the order shown. The major and minor version number are both mandatory. The minor version number must included, even if it is zero.
The revision number may be left off when it zero.
The build number is optional too, but may only be included if all the other three values are included. All missing values must be assumed to be zero.

The *major* version number signals a major new release, often available as an paid *upgrade*.
The *minor* version number version number signals a significant update of the same release,

often available as a free *update*. The *revision* number signals a lesser update, typically available as a free *hotfix*.
While the first public release of a product may have version number 1.0, earlier development releases should have version numbers too.

Each value consists at most 3 digits. No other characters are allowed. Zeroes are allowed, leading zeroes are not; `0.99` is a valid version number, and so is `1.0`, but `1.01` is not. Trailing zeroes must not be left out and aren't implied; version `2.1` and version `2.10` are two different versions, and version `2.1` comes before version `2.9`. The minimum version number is `0.1`. The maximum version number is `999.999.999.999`.

Build Number

The build number is a number that is different for each build of the same *major.minor.revision* version, and may indicate the platform the code was build for. Genealogy software developers that wish to distinguish between 32-bit and 64-bit builds *must not* do so by messing with the *major.minor.revision* version, but may opt to use build number `32` and build number `64` to indicate the 32-bit and 64-bit builds respectively. Systems that use the build number to indicate 32-bit and 64-bit build must always specify all four values.

Many developers use an ever-increasing build number for their product, and can continue doing so, but that product build number must not be used as part of the <PRODUCT_VERSION_NUMBER>. Developers that wish to include their product build number in a GEDCOM file may use it as part of the product name.

Microsoft uses the words build and revision differently; if your genealogy system is a Microsoft .NET application, do not worry too much about it, simply use *major.minor.build.revision* as defined within .NET, and you should be fine.

PUBLICATION_DATE:= {Size=10:11}

<DATE_EXACT>
The date this source was published or created.

RECEIVING_SYSTEM_NAME:= {Size=1:20}

The system identifier of the system expected to process the GEDCOM file.

GEDCOM Interpretation

All GEDCOM writers must default the HEAD.DEST value to their own system identifier.
The HEAD.DEST value specifies the interpretation of any GEDCOM extensions. In a typical GEDCOM file the HEAD.SOUR and HEAD.DEST values are identical; and the meaning of that typical situation is that the GEDCOM writer used its own GEDCOM extensions, if any.
A GEDCOM 5.5.5 writer should only specify another system identifier than its own, when the GEDCOM file is intended for import by a specific system, and the GEDCOM writer produces GEDCOM extensions for that system *instead of* its own. This is not a generally recommended practice, but can be handy when producing GEDCOM files for some particular third party system. While the HEAD.DEST value may be a system identifier for an old and no longer supported system, it may only be a system identifier for a system known to support GEDCOM 5.5.5.

Nonsense Values

GEDCOM specifications before the GEDCOM 5.5.1 Annotated Edition (2018 CE) failed to clearly specify the usage of the HEAD.DEST value. Many GEDCOM 5.5 and 5.5.1 files contain nonsense values such as `ANY`, `GEDCOM`, `GEDCOM55`, and `Other`, despite the fact

that the spec demands the value to be a system identifier.

GEDCOM 5.5.5 demands that the HEAD.DEST value is a real system identifier. Known nonsense values must not be used as system identifiers.

A GEDCOM 5.5.5 validator must issue a strong warning when the HEAD.SOUR and HEAD.DEST values aren't identical, and a fatal error when it recognises a known nonsense value.
A GEDCOM validator should also issue an error when it recognises a system identifier for a system that does not support GEDCOM 5.5.5, and a strong warning when it does not recognise the system identifier used (probable typo).

More information about system identifiers is provided in Chapter 4 GEDCOM System Identifiers, page 117.

RELATION_IS_DESCRIPTOR:= {Size=1:25}

A word or phrase that states object 1's **relation** is object 2. For example you would read the following as "Joe Jacob's high-school geography teacher is the individual identified by @I551@":

0 INDI
1 NAME Joe /Jacob/
1 ASSO @I551@
2 RELA high-school geography teacher

The <RELATION_IS_DESCRIPTOR> is the ASSO.RELA line value, which provides a description of the association created by the <<ASSOCIATION_STRUCTURE>> (ASSO record) it is part of.
Notice that the <RELATION_IS_DESCRIPTOR> *does not* provide any predefined values, but is a free-form text field; by the user, for the user.

Applications cannot and should not try to rely on the user or other applications using particular values, or any particular value having any particular meaning.
That said, some common values are: `godfather`, `godmother`, `attendant`, `informant`, `witness` and `Other`, but you may encounter many different values, such as `godparent`, `officiating priest`, including illegal values such as `brother`, or `Grand pere maternel` (maternal grandfather in French).

RELIGIOUS_AFFILIATION:= {Size=1:90}

A name of the religion with which this person, event, or record was affiliated.

RESPONSIBLE_AGENCY:= {Size=1:120}

The organization, institution, corporation, person, or other entity that has responsibility for the associated context. For example, an employer of a person of an associated occupation, or a church that administered rites or events, or an organization responsible for creating and/or archiving records.

ROLE_DESCRIPTOR:= {Size=1:25}

A word or phrase that identifies a person's role in an event being described. This should be the same word or phrase, and in the same language, that the recorder used to define the role in the actual record.

ROLE_IN_EVENT:= {Size=3:27}

[CHIL | HUSB | WIFE | MOTH | FATH | SPOU | **(** <ROLE_DESCRIPTOR> **)**]

Indicates what role this person played in the event that is being cited in this context. For example, if you cite a child's birth record as the source of the mother's name, the value for this field is "MOTH". If you describe the groom of a marriage, the role is "HUSB". If the

role is something different than one of the six relationship role tags listed above then enclose the role name within matching parentheses.

The six predefined values must not be within parentheses, but that additional, system-defined or user-defined values must be. It is an error to put any of the predefined values within parentheses, or not use parentheses for other values.

ROMANISATION_METHOD:= {Size=5:30}

[<user defined> | pinyin | romaji | wadegiles]
Indicates the method used in transforming the text to a romanised transcription.

SCHOLASTIC_ACHIEVEMENT:= {Size=1:248}

A description of a scholastic or educational achievement or pursuit.

SEX_VALUE:= {Size=1:1}

A code that indicates the sex of an individual:

M = Male
F = Female
X = Intersex
U = Unknown (not found yet)
N = Not recorded

U is the default value.

The values U and N are mostly used for stillborn children, but must be used for every situation where the sex is unknown (yet).
The difference between the two values is significant. The value U merely says that you do not know the sex (yet), because you did not find or consult a record. The value N says that the record did not record a sex, or that you are sure that there is no record.

The current GEDCOM specification allows just one sex per individual. A future GEDCOM version should provide the ability to specify different values for the biological, official and legal sex. The current field should be used to record the official sex, i.e. the sex on the birth record.

SOURCE_CALL_NUMBER:= {Size=1:120}

An identification or reference description used to file and retrieve items from the holdings of a repository.

SOURCE_DESCRIPTIVE_TITLE:= {Size=1:4095}

The title of the work, record, or item and, when appropriate, the title of the larger work or series of which it is a part.

For a *published* work, a book for example, might have a title plus the title of the series of which the book is a part. A magazine article would have a title plus the title of the magazine that published the article.

For An *unpublished* work, such as:

- A letter might include the date, the sender, and the receiver.
- A transaction between a buyer and seller might have their names and the transaction date.
- A family Bible containing genealogical information might have past and present owners and a physical description of the book.
- A personal interview would cite the informant and interviewer.

SOURCE_FILED_BY_ENTRY:= **{Size=1:60}**

This entry is to provide a short title used for sorting, filing, and retrieving source records.

SOURCE_JURISDICTION_PLACE:= **{Size=1:120}**

<PLACE_NAME>
The name of the lowest jurisdiction that encompasses all lower-level places named in this source. For example, "Oneida, Idaho" would be used as a source jurisdiction place for events occurring in the various towns within Oneida County. "Idaho" would be the source jurisdiction place if the events recorded took place in other counties as well as Oneida County.

SOURCE_MEDIA_TYPE:= **{Size=1:15}**

[audio | book | card | electronic | fiche | film | magazine | manuscript | map | newspaper | photo | tombstone | video]

A code, selected from one of the media classifications choices above, that indicates the type of material in which the referenced source is stored.

SOURCE_ORIGINATOR:= **{Size=1:255}**

The person, agency, or entity who created the record. For a published work, this could be the author, compiler, transcriber, abstractor, or editor. For an unpublished source, this may be an individual, a government agency, church organization, or private organization, etc.

SOURCE_PUBLICATION_FACTS:= **{Size=1:4095}**

When and where the record was created. For published works, this includes information such as the city of publication, name of the publisher, and year of publication.

For an unpublished work, it includes the date the record was created and the place where it was created. For example, the county and state of residence of a person making a declaration for a pension or the city and state of residence of the writer of a letter.

SUBMITTER_NAME:= **{Size=1:60}**

The name of the submitter formatted for display and address generation.

SYSTEM_ID:= **{Size=1:20}**

A system identification name. This name must be unique for each system (product), different from any other system. The name may include spaces, and is *not* restricted to ASCII characters.

The system identifier occurs as a line value of the mandatory HEAD.SOUR and HEAD.DEST records in the GEDCOM header.

TEXT:= **{Size=1:32767}**

A Unicode string.
Some rules and restrictions apply, see line_char definition in the GEDCOM grammar.

TEXT_FROM_SOURCE:= **{Size=1:32767}**

<TEXT>
A verbatim copy of any description contained within the source. This indicates notes or text that are actually contained in the source document, not the submitter's opinion about the source. This should be, from the evidence point of view, "what the original record keeper said" as opposed to the researcher's interpretation. The word TEXT, in this case, means from the text which appeared in the source record including labels.

TIME_VALUE:= **{Size=7:12}**

hh:mm[:ss[.fs]]
The time of a specific event, usually a computer-timed event, where:

hh = hours on a 24-hour clock, one or two digits; no leading zeroes {0:23}
mm = minutes, a two-digit value *with* leading zeroes {0:59}
ss = seconds, a two-digit value, *with* leading zeroes {0:59}
fs = a two-digit decimal fraction of a second {0:99}

USER_REFERENCE_NUMBER:= **{Size=1:20}**

A user-defined number or text that the submitter uses to identify this record. For instance, it may be a record number within the submitter's automated or manual system, or it may be a page and position number on a pedigree chart.

The <USER_REFERENCE_NUMBER> is the line value of the optional REFN record. The optional REFN occurs in the FAM, INDI, OBJE, NOTE, REPO and SOUR records; so, in all the <LINEAGE_LINKED_RECORD> types.

While the name of this line value strongly suggests that it should be a number, the line value is not limited to digits. It is free-form text that may include any character. It should really have been called <USER_REFERENCE_IDENTIFIER> instead of <USER_REFERENCE_NUMBER>.

The <USER_REFERENCE_NUMBER> allows users to associate any kind of number with these records. These record types allow any number of REFN subrecords, so users can associate as many identifiers to a record as they like.

The user-defined identifiers used need not be unique. The same identifier may be used multiple times, on different records.

The REFN record is for user-defined numbers, the IDNO record is for third-party numbers.

USER_REFERENCE_TYPE:= **{Size=1:40}**

A user-defined definition of the <USER_REFERENCE_NUMBER>.

This value is free-form text, but meant to be a value from a relatively short user-defined list. The <USER_REFERENCE_TYPE> is the line value of the optional REFN.TYPE record; the REFN line value specifies a user reference number, the REFN.TYPE specifies that number's type.
This allows users to classify the reference numbers they use in any way they see fit.

USER_TEXT:= **{Size=1:32767}**

Free-form user text. Comments, opinions.

WHERE_WITHIN_SOURCE:= **{Size=1:248}**

Specific location within the information referenced. For a published work, this could include the volume of a multi-volume work and the page number(s). For a periodical, it could include volume, issue, and page numbers. For a newspaper, it could include a column number and page number. For an unpublished source or microfilmed works, this could be a film or sheet number, page number, frame number, etc. A census record might have an enumerating district, page number, line number, dwelling number, and family number. The data in this field should be in the form of a label and value pair, such as Label1: value, Label2: value, with each pair being separated by a comma. For example, Film: 1234567, Frame: 344, Line: 28.

XREF:= **{Size=3:22}**

Either a pointer or an unique cross-reference identifier. If this element appears before the tag in a GEDCOM line, then it is a cross-reference identifier. If it appears after the tag in a

GEDCOM line, then it is a pointer.

The at signs that delimit a cross-reference identifier or pointer are part of that cross-reference identifier or pointer. The identifying part within the at signs has a minimum length of 1, and maximum length of 20 code units.

Legal Characters

Each <XREF> corresponds to a **xref_ID** in the GEDCOM grammar, the identifying part within the at signs corresponds to a **identifier_string**.
GEDCOM 5.5.1 and earlier allowed almost any character in an identifier string, with a special restriction regarding the use of the number sign (#), and restrictions regarding the use of colon (:) and exclamation mark (!) in an <XREF>.
An identifier string must not start with a number sign (#), as that might create confusion with an escape sequence prefix (@#), but the number sign is no longer legal in identifiers. The Lineage-Linked Form reserved the colon (:) and exclamation mark (!) future feature. They are a no longer reserved, but also no longer legal in identifiers. **GEDCOM 5.5.5 solves many pointer problems simply by restricting identifiers to alphanumeric characters**.

References

- GEDCOM Identifiers: Length

XREF:FAM:= **{Size=3:22}**

A pointer to, or a cross-reference identifier of, a <<FAM_GROUP_RECORD>> (FAM record).

XREF:INDI:= **{Size=3:22}**

A pointer to, or a cross-reference identifier of, an <<INDIVIDUAL_RECORD>> (INDI record).

XREF:NOTE:= **{Size=3:22}**

A pointer to, or a cross-reference identifier of, a <<NOTE_RECORD>> (NOTE record).

XREF:OBJE:= **{Size=3:22}**

A pointer to, or a cross-reference identifier of, a <<MULTIMEDIA_RECORD>> (OBJE record).

XREF:REPO:= **{Size=3:22}**

A pointer to, or a cross-reference identifier of, a <<REPOSITORY_RECORD>> (REPO record).

XREF:SOUR:= **{Size=3:22}**

A pointer to, or a cross-reference identifier of, a <<SOURCE_RECORD>> (SOUR record).

XREF:SUBM:= **{Size=3:22}**

A pointer to, or a cross-reference identifier of, a <<SUBMITTER_RECORD>> (SUBM record).

YEAR:= **{Size=3:4}**

A numeric representation of the calendar year in which an event occurred.

One and two-digit years are illegal. Years 1 through 99 must be padded out to at least 3 digits by using leading zeroes.

> **Three Digits Demand is Obsolete**
>
> The demand to use at least 3 digits for the year is obsolete. This demand was put in to avoid possible confusion between years and days as entered into early genealogy applications.
> Date entry in genealogy applications continues to require developer and user attention, particularly in locales still using antilogical date formats. However, the fixed order date formats used in GEDOM do not allow confusion between days and years.

The GEDCOM File

File Name

The GEDCOM file is normally created in a user directory. The filename extension is `.GED` or `.ged`; the casing does not really matter, but by convention the filename extension is either all-uppercase or all-lowercase.
Macintosh filenames do not use file extensions.

Multi-Volume GEDCOM Files

GEDCOM 5.0 (1991) through 5.5.1 support multi-volume GEDCOM files. A multi-volume file is a file split into multiple files (the volumes); each of the volumes should fit onto a single floppy disk or diskette, even if the original file does not. This feature was never necessary, as multi-volume archivers were widely available.

In practice, all the Multi-volume GEDCOM support accomplished is needlessly complicating the reading of GEDCOM files; a GEDCOM 5.5.1 reader must not assume that the end of a file is the end of the GEDCOM file; the GEDCOM file may continue on another volume (the next file in the multi-volume series).
The *GEDCOM 5.5.1 Annotated Edition* deprecated this misfeature and the GEDCOM 5.5.5 specification obsoletes it; it is no longer allowed to break a GEDCOM file into multiple files. This simplifies GEDCOM readers; the end of the GEDCOM file is the end of the GEDCOM file.

User-Defined Tags

Use of user-defined GEDCOM tags is not encouraged. Applications requiring the use of nonstandard tags must define them with a leading underscore so that they will not conflict with future GEDCOM standard tags. The meaning and interpretation of user-defined tags is system-dependant. The system that defined the tags is identified in HEAD.DEST.

The system in HEAD.DEST is often, but not always, the same system as that in HEAD.SOUR; applications can and do generate GEDCOM files intended for particular third-party systems.

Escape Sequences

Use of GEDCOM escape sequences is not recommended. The only use of escape sequences in the current Lineage-Linked Form is the use of calendar escape sequences, to specify a calendar and a date in single line value.

References

- Multi-volume GEDCOM files

Sample Lineage-Linked GEDCOM File

The example below is a sample GEDCOM file of genealogical information about three individuals who are members of the same family — father, mother, and child. In the example, "Joe/Williams/" is the value specified by the tag NAME under the INDI tag for the record (@I3@). Other values in other lines, such as the birth date and place, provide additional information about Joe Williams. The value (@F1@) specified by the FAMC tag is a pointer to the <<FAM_GROUP_RECORD>> (@F1@) of which Joe Williams is a child. Included also in this GEDCOM file example are three other record types: a source record, a submitter record, and a repository record. These records are pointed to from within other records in the GEDCOM file. This shows how pointer values can be used in creating a Lineage-Linked GEDCOM file.

Example:

```
0 HEAD
1 GEDC
2 VERS 5.5.5
2 FORM LINEAGE-LINKED
3 VERS 5.5.5
1 CHAR UTF-8
1 SOUR GS
2 NAME GEDCOM Specification
2 VERS 5.5.5
2 CORP gedcom.org
3 ADDR
4 CITY LEIDEN
3 WWW www.gedcom.org
1 DATE 2 Oct 2019
2 TIME 0:00:00
1 FILE 555Sample.ged
1 LANG English
1 SUBM @U1@
0 @U1@ SUBM
1 NAME Reldon Poulson
1 ADDR
2 ADR1 1900 43rd Street West
2 CITY Billings
2 STAE Montana
2 POST 68051
2 CTRY United States of America
1 PHON +1 (406) 555-1232
0 @I1@ INDI
1 NAME Robert Eugene /Williams/
2 SURN Williams
2 GIVN Robert Eugene
1 SEX M
1 BIRT
2 DATE 2 Oct 1822
2 PLAC Weston, Madison, Connecticut, United States of America
2 SOUR @S1@
3 PAGE Sec. 2, p. 45
1 DEAT
2 DATE 14 Apr 1905
2 PLAC Stamford, Fairfield, Connecticut, United States of America
```

1 BURI
2 PLAC Spring Hill Cemetery, Stamford, Fairfield, Connecticut, United States of America
1 FAMS @F1@
1 FAMS @F2@
1 RESI
2 DATE from 1900 to 1905
0 @I2@ INDI
1 NAME Mary Ann /Wilson/
2 SURN Wilson
2 GIVN Mary Ann
1 SEX F
1 BIRT
2 DATE BEF 1828
2 PLAC Connecticut, United States of America
1 FAMS @F1@
0 @I3@ INDI
1 NAME Joe /Williams/
2 SURN Williams
2 GIVN Joe
1 SEX M
1 BIRT
2 DATE 11 Jun 1861
2 PLAC Idaho Falls, Bonneville, Idaho, United States of America
1 FAMC @F1@
1 FAMC @F2@
2 PEDI adopted
1 ADOP
2 DATE 16 Mar 1864
0 @F1@ FAM
1 HUSB @I1@
1 WIFE @I2@
1 CHIL @I3@
1 MARR
2 DATE Dec 1859
2 PLAC Rapid City, Pennington, South Dakota, United States of America
0 @F2@ FAM
1 HUSB @I1@
1 CHIL @I3@
0 @S1@ SOUR
1 DATA
2 EVEN BIRT, DEAT, MARR
3 DATE FROM Jan 1820 TO DEC 1825
3 PLAC Madison, Connecticut, United States of America
2 AGNC Madison County Court
1 TITL Madison County Birth, Death, and Marriage Records
1 ABBR Madison BMD Records
1 REPO @R1@
2 CALN 13B-1234.01
0 @R1@ REPO
1 NAME Family History Library
1 ADDR
2 ADR1 35 N West Temple Street
2 CITY Salt Lake City
2 STAE Utah
2 POST 84150
2 CTRY United States of America
0 TRLR

Chapter 3 Using Character Sets in GEDCOM

Introduction

GEDCOM was created to exchange genealogical data in multiple languages, so it needs to support the characters used by those languages. Back in the 20th century of the Common Era, when GEDOM was still focussed on Western languages, FamilySearch wisely decided to disallow code pages and use ANSEL instead. GEDCOM 5.3 (1993 CE) specification and later allow Unicode in support of non-Western languages, with ANSEL remaining the default. The GEDCOM 5.4 (1995 CE), GEDCOM 5.5 and GEDCOM 5.5.1 (1999 CE) specification all state that Unicode will eventually replace ANSEL as the standard character set for GEDCOM.
In the twenty years that passed since the release of GEDCOM 5.5.1 the industry has switched to Unicode. The *GEDCOM 5.5.1 Annotated Edition* (2018 CE) already states all genealogy applications should be Unicode-based and always export to Unicode. GEDCOM 5.5.5 completes the transition to Unicode.
GEDCOM 5.5.5 does not allow ANSEL. GEDCOM 5.5.5 demands Unicode
Systems should continue to support the ANSEL encoding for importing GEDCOM 5.5.1 files.

GEDCOM Character Sets and Encoding

GEDCOM 5.5.5 has just one legal character set: Unicode. Earlier version of GEDCOM supports three character sets: ASCII, ANSEL and Unicode.
ASCII and ANSEL are characters sets with just one encoding. Unicode has multiple encodings, GEDCOM supports the two most important ones, UTF-8 and UTF-16.
Thus, GEDCOM supports four character encodings: ASCII, ANSEL, UTF-8 and UTF-16. The `HEAD.CHAR` line values specifying these encodings are `ASCII`, `ANSEL`, `UTF-8`, and `UNICODE`.

UTF-16: `UNICODE`

Previous GEDCOM specifications erroneously refer to UTF-16 as UNICODE (all-caps), and introduced the `HEAD.CHAR` line value `UNICODE` for it.
For compatibility's sake, the `HEAD.CHAR` value for UTF-16 remains `UNICODE`. The `UNICODE` value must be understood as meaning UTF-16, and *must not* be used for any other Unicode encoding. UTF-8 GEDCOM files *must* use the `HEAD.CHAR` value `UTF-8`.

Legal Encodings

GEDCOM 5.5.5 has just two legal encodings, UTF-8 and UTF-16. Earlier versions of GEDCOM support four legal encodings: ASCII, ANSEL, UTF-8 and UTF-16.
Historically, many system have created not-really-GEDCOM files using illegal character sets and encodings. Developers want users to be able to import third-party GEDCOM and not-really-GEDCOM files, so many GEDCOM 5.5 and 5.5.1 readers support the use of these illegal character sets and encodings. That well-intentioned flexibility has a serious deleterious effect. By obviating the need for the developers of the offending third-party product to fix their GEDCOM output, those too-flexible GEDCOM readers keep the malpractice and the problems its creates alive.
GEDCOM 5.5.5 ends this malpractice without breaking existing software.

GEDCOM 5.5.5 does not tolerate the use of illegal character sets or encodings.

- GEDCOM 5.5.5 writers *must not* use anything but the legal encodings.
- GEDCOM 5.5.5 readers *must not* accept anything but the legal encodings.
- GEDCOM 5.5.5 readers *must* reject all illegal encodings.

Systems that currently accept illegal character sets and encodings in GEDCOM 5.5.1 or earlier may continue to do so for the sake of compatibility, but only in GEDCOM 5.5.1 and earlier. Developers that want to continue using illegal character sets and encodings, must continue to do

so using GEDCOM 5.5.1 or earlier.

Support

All systems supporting GEDCOM 5.5.5 (or later) should default GEDCOM export to GEDCOM 5.5.5.
All GEDCOM 5.5.5 export *must* use the Unicode character set. All Western systems should default their GEDCOM export to UTF-8 encoding. All Eastern systems should default their GEDCOM export to UTF-16 encoding. Users should be able to override that default by setting a file property.

Export of Unicode data to ASCII or ANSEL would be practically sure to lose information. That is why GEDCOM 5.5.5 does not allow it. Unicode-based systems that already support export to ANSEL-encoded GEDCOM 5.5.1 may and should continue to offer that option, for the sake of compatibility with the user's existing practices. However, export to non-Unicode encodings should not be offered via the regular GEDCOM export dialog box, but only as an advanced option.

Real World UTF-8 only Export

Several applications were offering Unicode-only GEDCOM export before GEDCOM 5.5.5 already. RootsMagic has offered nothing but UTF-8 GEDCOM export since it became Unicode-based with RootsMagic 4 in 2008 CE. Software MacKiev Family Tree Maker changed to UTF-8 only export starting with Family Tree Maker 2017, and Ahnenblatt changed to UTF-8 only export with Ahnenblatt 3.0, in 2019 CE.

Systems should be able to import all four legal encodings.
UTF-8 and UTF-16 are the preferred encodings of GEDCOM files. GEDCOM 5.5.5 systems *must not* create ANSEL GEDCOM files, but import of ANSEL GEDCOM 5.5.1 files must be supported for backward compatibility.

Code Page-based Systems

GEDCOM 5.5.5 demands a Unicode-based system.
Code page-based systems should no longer be in use. Users of a code page-based system are strongly recommended to stop using it and upgrade to one of the many Unicode-based system.

Code page-based systems *must not* claim or attempt to support GEDCOM 5.5.5, but must continue to use GEDCOM 5.5.1 or earlier.
The rule for code page-based systems has always been to export to GEDCOM using ANSEL, but it is recommended that even code-page based system now default their GEDCOM export to using UTF-8.

ANSEL

ANSEL (American National Standard for Extended Latin Alphabet Coded Character Set for Bibliographic Use, Z39.47-1985) is an 8-bit character set. The official specification can be obtained from the American National Standards Institute (ANSI).
ANSEL was *the* American Library Association character set, and was used in library systems worldwide, including the MARC (Machine-Readable Catalog) format. Those systems have now transitioned to Unicode.

ANSEL is a superset of ASCII. ANSEL extends ASCII with a some useful characters and non-spacing character modifiers (diacritics), resulting in a single character set for Western languages (instead of a confusing mess of incompatible code pages).

ANSEL versus “ANSEL”

FamilySearch has extended ANSEL and kept calling the resulting character set ANSEL, in direct

violation of the ANSEL specification. FamilySearch also kept changing what's in their “ANSEL”; different versions of GEDCOM have different “ANSEL” tables. GEDCOM 5.5.5 provides a consolidated “ANSEL” table in Appendix C ANSEL Character Set. It is strongly recommended that genealogy software developers use this single table for all GEDCOM versions they support.

Previous versions of GEDCOM stated that ANSEL is the preferred character encoding for GEDCOM files. GEDCOM 5.5.1 (1999 CE) already stated that this would shortly change to UTF-8 and UTF-16. GEDCOM 5.5.1 remained the last version of GEDCOM for almost two decades and during that time, the industry has switched to Unicode as the preferred character set and UTF-8 as the preferred encoding. GEDCOM 5.5.5 merely makes that official.

Starting with GEDCOM 5.5.5, Unicode is the preferred character set for GEDCOM files. Systems can use either the UTF-8 or UTF-16 encoding.
Use of ANSEL remains legal, but is deprecated now.

ASCII

The American Standard Code for Information Interchange (ASCII) is one of the oldest character set standards (1963 CE). ASCII is an USA-centric character set. Most Western languages need characters not supported by ASCII, which is why ANSEL was the preferred character set for GEDCOM files. Many languages need characters not supported by ANSEL, which is why Unicode is the preferred character set for GEDCOM files. ANSEL and Unicode are both supersets of ASCII.

Previous versions of GEDCOM referred to in various incorrect and confused ways, such as “ASCII (USA Version)”; There are no country-specific versions of ASCII. There is just ASCII, which is American. Many country-specific code pages are based on ASCII, but *are not ASCII*.

Previous versions of GEDCOM also referred to ASCII as “8-bit ASCII”, and that is a contradiction in terms. There is no such thing as 8-bit ASCII. ASCII is a 7-bit character set.

Unicode

Unicode and ISO 10646 are not exactly the same thing, but do define the same character set. It is a rather persistent myth that Unicode is a 16-bit character set - it is not. Unicode is a 21-bit character set that currently supports more than a quarter million different characters.
Unicode *is* a superset of ASCII and the many 8-bit code pages it replaces.

Unicode is a single character set that supports quite a few different encodings. GEDCOM allows only the two most important ones, UTF-8 and UTF-16. Support for UTF-8 and UTF-16 is built into all modern operating systems; developers need not and should not create their own conversions.

Developer who ignore the above advice and try to make their own conversions anyway should take particular care to make sure that their conversion from the UTF-16 encoding used by the operating system to UTF-8 is indeed a conversion to UTF-8 and not a conversion to CESU-8. GEDCOM readers that take advantage of operating system routines are not unlikely to reject ostensible UTF-8 GEDCOM files that are actually CESU-8 encoded because of illegal code sequences. This is a security measure.

To understand Unicode and the supported encodings, it is essential to understand the difference between characters, code points and code units.
Very briefly: A code point is a particular value within a character set. A single character may consist of multiple code points.
A code unit is the smallest size used in a particular encoding of a character set; UTF-8 has 8-bit code units, UTF-16 has 16-bit code units. An encoding may require multiple code units to represent one code point.

Byte Order Mark

GEDCOM files are text files, so UTF-8 GEDCOM files are UTF-8 text files. Per Unicode rules,

starting the text file with Byte Order Mark (BOM) is mandatory for UTF-16 text files, but optional for UTF-8 text files.

Starting UTF-8 GEDCOM files with a BOM makes sure that text editors will recognise the GEDCOM file as UTF-8 encoded, so that they do not misrecognise the encoding and mess up the file. Starting UTF-8 GEDCOM files with a BOM prevents really old genealogy applications, that do not understand UTF-8, from recognising the GEDCOM file as a GEDOM file. Many old applications do not check the encoding, and will try to import the GEDCOM file anyway, despite not supporting the encoding used (and not supporting Unicode at all), resulting in significantly mangled data and data loss. Including a BOM prevents these old applications from doing so.

The *GEDCOM 5.5.1 Annotated Edition* strongly recommends that UTF-8 GEDCOM files start with a BOM. **GEDCOM 5.5.5 specifies that UTF-8 GEDCOM files *must* start with a Byte Order Mark (BOM).** This rule aligns with existing practice, as started with FamilySearch PAF 5.0 (2000 CE).

Using Character Encodings

Previous versions of GEDCOM suggested that it might be possible to change character sets in the middle of a GEDCOM file - it is not. That notion was abandoned, but even GEDCOM 5.5.1 still contained confusing remnants of that notion. Each GEDCOM file uses just one encoding throughout.
Previous versions of GEDCOM suggested that, regardless of whatever encoding is used for the GEDCOM file, the GEDCOM header should always be ANSEL-encoded. That is wrong. The entire GEDCOM file, including the GEDCOM header, always uses just one encoding.
The encoding used by a GEDCOM file is specified in the GEDCOM header through the mandatory `HEAD.CHAR` record.

The following partial GEDCOM header provides an example:

```
0 HEAD
1 SOUR PAF
2 VERS 5.2.18
1 DEST PAF
1 CHAR UTF-8
...
```

Detecting Character Encoding

The GEDCOM specification provides practically no information about detecting GEDCOM files and the encoding used by GEDCOM files.

References

- The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)
- 0 HEAD Value
- GEDCOM & FTW TEXT Magic
- GEDCOM Magic Values
- GEDCOM Character Encodings

Unicode Books

The Unicode Standard has seen several editions since the publication of GEDCOM 5.5.1

back in 1999. It is a reference, not very suitable for learning about Unicode.
The Unicode Consortium web site offers many helpful resources, including an introduction to Unicode and a FAQ.

Two useful books are:

- *Unicode Demystified; A Practical Programmer's Guide to the Encoding Standard* by Richard Gillam (2003, Addison-Wesley, pp. 853, ISBN 0-201-70052-2)
- *Developing International Software, Second Edition* by Dr. International (2003, Microsoft Press, pp. 1060, ISBN 0-6536-1583-7)
  This is the revised edition of *Developing International Software for Windows 95 and Windows NT* by Nadine Kano (1995, Microsoft Press, pp. 743, ISBN 1-55615-840-8).

Chapter 4 GEDCOM System Identifiers

System Identifier Registration

Each system that reads or writes GEDCOM files must have a system identifier to identify that system by. That system identifier must be unique, so that there can be no confusion about which system is meant.
The obvious way to ensure uniqueness is through a centrally maintained registry of system identifiers. FamilySearch used to have a registration process, but there is no registry today or registration process now. FamilySearch silently abandoned GEDCOM around 2000 CE. FamilySearch does not provide a list of system identifiers registered before 2000 CE, and most current genealogy applications were created after 2000 CE.

GEDCOM Header

System identifiers are used in the GEDCOM header, to identify the system that created the GEDCOM file, and the system that the GEDCOM file is for. The HEAD.SOUR value identifies the system that created the GEDCOM file, the HEAD.DEST value identifies the system that it is for. It is the HEAD.DEST value, not the HEAD.SOUR value, that identifies the meaning and interpretation of any GEDCOM extensions used in the GEDCOM file.

A typical GEDCOM file isn't created for a specific other system, but does contain GEDCOM extensions specific to the system that created it; that's why the HEAD.DEST value must default to system's own system identifier. Thus, in a typical GEDCOM file, the HEAD.SOUR value and the HEAD.DEST value are identical.

Maximum System Identifier Length

The maximum length of a system identifier is 20 code units, as stated in the <SYSTEM_ID> definition.
Some confusion was created by the GEDCOM 5.5.1 chapter *GEDCOM Product Registration*, which stated that system identifiers may be up to 40 characters long. The actual definition takes precedence over statements in chapters, and that chapter is gone now.

Case-Insensitive

System identifiers aren't controlled line values and should be case-sensitive, but long existing practice is to treat system identifiers as case-insensitive. Therefore, GEDCOM System Identifiers are case-insensitive.
Developers are still advised to pick one particular casing and stick with it, for consistency's sake. Use of mixed case is preferred, but abbreviations should be uppercase.

Spaces

It is legal to use spaces in system identifiers. Use of spaces within system identifiers has always been allowed, despite earlier versions of GEDCOM saying otherwise, as FamilySearch themselves has been using system identifier that contain spaces for decades.

Non-ASCII

The use of non-ASCII characters in the system identifier is legal, and *not* discouraged.
Systems reading GEDCOM 5.5.5 files know the character set and encoding used in the header as soon as they've processed the Byte Order Mark (BOM).

Digits

System identifiers may contain digits. However, system identifiers *must not* contain version numbers.

Version Numbers

System identifiers *must not* contain version numbers. Different versions of the same system must use the same system identifier. The version number of the system can and must be specified in HEAD.SOUR.VERS.

Editions

Some products are available in multiple editions, such as a Lite Edition or a Pro Edition. System identifiers *must not* indicate an edition. Different editions of the same system must use the same system identifier. The edition may be included in the product name HEAD.SOUR.NAME.

Platforms

The system identifier *should not* include the platform name (such as “Windows”).
Multiplatform systems *must not* include the platform name, but must use the GEDCOM reading and writing code, and the same system identifiers on all platforms.

Multiple Identifiers

Each system must be known by just one system identifier. Use of multiple, different identifiers for a single system is not allowed.

Some genealogy software developers have sold the same product under different names in different language regions, and then used a different name for each regional version. This is wrong.
The system identifier must not change merely because the user interface got translated.

Invalid System Identifiers

Systems creating GEDCOM 5.5.5 files *must* use a valid system identifier. Systems reading GEDCOM 5.5.5 files *must* refuse invalid system identifiers, even if they accept these when used in GEDCOM 5.5.1 files.
GEDCOM 5.5.5 readers *must* reject system identifiers that are too long, contain product version numbers or product editions.
GEDCOM 5.5.5 readers *should* also reject all otherwise valid system identifiers that are part of a collection of multiple system identifiers for a single system.

Nonsense Values

GEDCOM specifications before the GEDCOM 5.5.1 Annotated Edition (2018 CE) failed to clearly specify the usage of the HEAD.DEST value. Many GEDCOM 5.5 and 5.5.1 files contain nonsense values such as `ANY`, `GEDCOM55`, and `Other`, despite the fact that the specification demands that the HEAD.DEST value be a system identifier.

GEDCOM 5.5.5 does not tolerate this malpractice. GEDCOM 5.5.5 demands that the HEAD.DEST value is a real system identifier. Known nonsense values must not be used as system identifiers.

Known Nonsense Values

The following are known nonsense values for the system identifier that *must not* be used. GEDCOM 5.5.5 readers that encounter any of these values instead of an actual system identifier, *must* reject the file as not-GEDCOM, specifically as not even having a valid GEDCOM header.

known nonsense values
`ANY`
`GED55`
`GEDCOM`
`GEDCOM55`
`Other`

The above table of known nonsense values may be expanded as additional nonsense values are identified. Developers are encouraged to take full advantage of the table published in the latest GEDCOM version.
Obviously, none of the known nonsense values may be adopted as the system identifier for any new or existing system.

Quick summary of System Identifier Rules

- A system identifier must be unique
- System identifiers are case-insensitive.
- Spaces are allowed
- Digits are allowed
- Non-ASCII characters are allowed
- The maximum length is 20 code units
- Same, single system identifier for all versions and platforms
- Version numbers or version indicators are not allowed
- The system identifier must not include any platform indication
- Known nonsense values are illegal

GEDCOM Reader Rules

- Treat system identifiers as case-insensitive
- Accept spaces in system identifiers
- Accept non-ASCII characters (i.e. figure out the character encoding first)
- Issue a fatal error (not even a correct GEDCOM header!) when the system identifier is too long
- Reject known nonsense value with a fatal error: invalid GEDCOM header

GEDCOM Validator Best Practice

- Allow spaces, inform that spaces were officially illegal, never really illegal
- Warn against usage of non-ASCII characters in pre-GEDCOM 5.5.5 files only
- Issue a fatal error (not even a correct GEDCOM header!) when the system identifier is too long
- Issue a fatal error (not even a correct GEDCOM header!) when encountering known nonsense values
- Upon encountering an ALL-CAPS system identifier, suggest the use of lower case for readability
- Try to detect version numbers in system identifiers, and issue a strong warning against it

References

- GEDCOM System Identifiers
- GEDCOM SOUR and DEST

Appendix A Lineage-Linked GEDCOM Tag Definition

Introduction

Appendix A is a glossary of the tags used in the Lineage-Linked Form. These tags are used in a hierarchical structure to describe individuals in terms of their families, names, dates, places, events, roles, sources, and relationships. Control information and other kinds of data intended for computer processing is also included.

.

Extending the Form

Records defined in the Lineage-Linked Form cannot not be redefined.
It is legal to extend the Lineage-Linked Form, but only by using user-defined tags which must begin with an underscore. The meaning of GEDCOM tags is defined by their context, e.g. HEAD.SOUR and FAM.SOUR are two different kind of sources. Software developers *are* allowed to introduce _TAG.SOUR; as that does not redefine any existing SOUR tag in any way. However, developers are disallowed from directly extending the Lineage-Linked Form using existing tags, so they are not allowed to introduce say OBJE.SOUR, but would have to use OBJE._SOUR instead.

A GEDCOM reader *should not* assume that predefined tags occur only in predefined context, but only assume that the records defined in the Lineage-Linked Form are as defined.

Illegal Tags

User-defined tags that do not start with an underscore are illegal.
GEDCOM readers that come across an illegal tag should issue a fatal error and abort. It may seem harsh to reject a GEDCOM file for one illegal tag, but there simply is no reason for a GEDCOM file to contain illegal tags. It's a trivial effort for the developer of the offending application to use underscores.

The obvious exception to this rule is a new version of the offending application reading an GEDCOM file produced by an earlier version of that application; applications should be able to read files produced by earlier versions of the same application, even if these files should otherwise be rejected.

FTW TEXT

The list of definitions below gives both the tag name and, within curly brackets, a longer text. Typically, the longer text is a valid English word, or two words connected by an underscore, and the tag an abbreviation of that word or phrase.

GEDCOM uses the tags, most of which are four code units long. Exactly one genealogy application decided to use the text within the curly brackets instead. That application is Family Tree Maker for Windows (FTW), and those files are not GEDCOM files. These file are known as FTW TEXT files.

Back when Family Tree Maker for Windows (Classic) was quite popular, these files used to be quite prevalent. There are few FTW TEXT files around any more. They can be converted to GEDCOM files using a freely downloadable copy of Family Tree Maker 2005 Starter Edition. A download link to the Family Tree Maker 2005 Starter Edition is provided in the Family Tree Maker support article *Opening Old and Unsupported Files in FTM 2008-2017 for Windows*. So, there is no need to support

FTW TEXT in modern genealogy applications.

FTW TEXT Detection

Detecting FTW TEXT files is easy.
While the first line of a GEDCOM file is `0 HEAD`, the first line of an FTW TEXT is `0 HEADER`.

GEDCOM Reader Best Practice

- do *not* add support for FTW TEXT
- **do detect** FTW TEXT, and when detected
  - inform the user that it isn't a GEDCOM file, but an FTW TEXT file
  - advise the user to convert the FTW TEXT to GEDCOM using Family Tree Maker 2005 Starter Edition
- if you *do* support FTW TEXT
  - do not do so silently, do not let the user keep thinking it's a GEDCOM file when it isn't
  - clearly state that you detected FTW TEXT, and are processing their FTW TEXT file
  - make sure you support both GEDCOM and FTW TEXT, but not FGTEWDTCEOXMT (a mixture of both); keep detecting illegal tags

A download link for Family Tree Maker 2005 Starter Edition is available in the MacKiev support item *Family Tree Maker support: Opening Old and Unsupported Files in FTM 2008-2017 for Windows*.

References

- FTW TEXT
- Family Tree Maker support: Opening Old and Unsupported Files in FTM 2008-2017 for Windows

Lineage-Linked GEDCOM Tag Definitions

This section provides the definitions of the Lineage-Linked GEDCOM tags and shows their formal name inside of the {braces}. The formal names are *not* used in place of the tag. Full understanding must come from the context in which the tag is used.

ABBR {ABBREVIATION}:=
A short name of a title, description, or name.

ADDR {ADDRESS}:=
The contemporary place, usually required for postal purposes, of an individual, a submitter of information, a repository, a business, a school, or a company.

ADR1 {ADDRESS1}:=
The first line of an address.

ADR2 {ADDRESS2}:=
The second line of an address.

ADR3 {ADDRESS3}:=
The third line of an address.

ADOP {ADOPTION}:=

Adoption is a legal event that changes a child's legal parents from one set of parents to another set of parents.
While some of the parents involved are likely to be biological or official parents, neither assumption should be made.

Adoption is an event that changes who the legal parents are. A child that has been adopted can be adopted again. The official parents should not be assumed to be the biological parents. A child can *and often is* adopted by a biological or official parent. In many jurisdictions, a child is technically always adopted by a couple, even if one of them is already is a legal parent.

AGE {AGE}:=

The age of the individual at the time an event occurred, or the age listed in the document.

AGNC {AGENCY}:=

The institution or individual having authority and/or responsibility to manage or govern.

ANUL {ANNULMENT}:=

Declaring a marriage void from the beginning (retroactively invalid).

ASSO {ASSOCIATES}:=

An indicator to link friends, neighbours, or associates, who aren't close relatives of an individual.

AUTH {AUTHOR}:=

The name of the individual who created or compiled information.

BAPM {BAPTISM}:=

The event of baptism, performed in infancy or later. (See also CHR, page 124.)

BARM {BAR_MITZVAH}:=

The religious ceremony held when a Jewish boy reaches age 13.

BASM {BAS_MITZVAH}:=

The religious ceremony held when a Jewish girl reaches age 13, also known as "Bat Mitzvah."

BIRT {BIRTH}:=

The emergence of offspring from their mother as a separate being.
Birth does not imply life. Birth includes stillbirth.

BURI {BURIAL}:=

The action of burying a body.
BURI includes all forms of burial, including burial at sea, and as there is no separate event for interment (entombment), BURI is used for that too.

CALN {CALL_NUMBER}:=

The number used by a repository to identify the specific items in its collections.

CAST {CASTE}:=

The name of an individual's rank or status in society which is sometimes based on racial or religious differences, or differences in wealth, inherited rank, profession, occupation, etc.

CAUS {CAUSE}:=

A description of the cause of the associated event or fact, such as the cause of death.

CENS {CENSUS}:=

The event of the periodic count of the population for a designated locality, such as a national or state Census.

CHAN {CHANGE}:=
Indicates a change, correction, or modification. Typically used in connection with a DATE to specify when a change in information occurred.

CHIL {CHILD}:=
The biological, official or legal (adopted) child of a parent or parents.

CHR {CHRISTENING}:=
The religious event of baptising *and* naming a child.

CHRA {ADULT_CHRISTENING}:=
The religious event of baptizing *and* naming an adult person.

CITY {CITY}:=
A lower level jurisdictional unit. Normally an incorporated municipal unit.

CONF {CONFIRMATION}:=
The religious rite that confirms membership of a church (*confirms* because previously established by baptism).

COPR {COPYRIGHT}:=
A statement that accompanies data to protect it from unlawful duplication and distribution.

CORP {CORPORATE}:=
A name of an institution, agency, corporation, or company.

CREM {CREMATION}:=
Disposal of a body by fire, by burning it to ashes.

CTRY {COUNTRY}:=
The name of the country.

DATA {DATA}:=
Data.

DATE {DATE}:=
The time of an event in a calendar format.

DEAT {DEATH}:=
The end of a life.

DEST {DESTINATION}:=
A system receiving data.

DIV {DIVORCE}:=
The legal dissolution of a marriage.

DIVF {DIVORCE_FILED}:=
An event of filing for a divorce by a spouse.

DSCR {PHYSICAL_DESCRIPTION}:=
The physical characteristics of a person, place, or thing.

EDUC {EDUCATION}:=
Indicator of a level of education attained.

EMAIL {EMAIL}:=
An electronic mail address.
GEDCOM 5.5.1 lists EMAIL as EMAI (no L) in this Appendix. A forgiving GEDCOM 5.5.1 reader may treat EMAI as a synonym for EMAIL.
A GEDCOM 5.5.5 writer *must* use EMAIL (the actual tag). A GEDCOM 5.5.5 reader *must* reject EMAI (no L) as an illegal tag and abort processing.

EMIG {EMIGRATION}:=

An event of leaving one's homeland with the intent of residing elsewhere.

ENGA {ENGAGEMENT}:=

An event of recording or announcing an agreement between two people to become married.

EVEN {EVENT}:=

Pertaining to a noteworthy happening related to an individual, a group, or an organisation.
An EVEN (event) structure is usually qualified or classified by a subordinate use of the TYPE record.

FACT {FACT}:=

Pertaining to a noteworthy attribute or fact concerning an individual, a group, or an organisation.
A FACT structure is usually qualified or classified by a subordinate use of the TYPE record.

FAM {FAMILY_GROUP}:=

The FAM (family group) structure records a single family group; a couple and their children. The group consist of two partners, either or both of which may be unknown, with or without children. The partners may or may not be spouses, and may or may not have children, but are biological, official or legal parents to each of the children in the group.
Recording a single family often requires more than one family group record.

FAMC {FAMILY_CHILD}:=

Identifies the family group in which an individual appears as a child.

FAMS {FAMILY_SPOUSE}:=

Identifies the family group in which an individual appears as a partner.

The name and abbreviation of this record are misleading: the individual need not be a spouse in that family group. The family group record is used for all relationships, not just marriages. *Do not assume* that the person is a spouse in that family group.

FAX {FACSIMILE}:=

Electronic facsimile transmission.

FCOM {FIRST_COMMUNION}:=

Literally the first communion an individual partakes in. Communion is a rite within christian churches, and the first communion is considered a rite of passage.

FILE {FILE}:=

The name of an external file, or, in the case of HEAD.FILE, the original filename of this GEDCOM file.

FONE {PHONETIC}:=

A phonetic rendering of a superior text string.

GIVN {GIVEN_NAME}:=

A given or earned name used for official identification of a person.

GRAD {GRADUATION}:=

An event of awarding educational diplomas or degrees to individuals.

HUSB {HUSBAND}:=

A partner in a FAM (family group) record, often male, often partner to a woman, and a biological, official or legal parent to each of the children of the couple.
The name of this record strongly suggests that the line value must identify a husband, but that is not the case; the relationship need not be a marriage, and the individual need not be male, it may identify a woman in a lesbian relationship.

References

- Same-Sex Marriage in GEDCOM

IDNO {IDENT_NUMBER}:=

An identifier, often called a number, assigned to identify a person within some significant external system.
The value typically isn't number, but a value containing spaces and dashes in addition to letters and digits.
The prime example is a passport “number”. American genealogists often record Social Security Numbers.

The difference between the INDO record and the REFN record is that the IDNO record is for third-party numbers, and the REFN record is for user-defined numbers.

IMMI {IMMIGRATION}:=

An event of entering into a new locality with the intent of residing there.

INDI {INDIVIDUAL}:=

A person.

LANG {LANGUAGE}:=

The name of the language used in a communication or transmission of information.

LATI {LATITUDE}:=

Latitude of a position on the globe.

LONG {LONGITUDE}:=

Longitude of a position on the globe.

MAP {MAP}:=

Pertains to a representation of measurements usually presented in a graphical form.

MARB {MARRIAGE_BANN}:=

An event of an official public notice given that two people intend to marry.

MARC {MARR_CONTRACT}:=

An event of recording a formal agreement of marriage, including the prenuptial agreement in which marriage partners reach agreement about the property rights of one or both, securing property to their children.

MARL {MARR_LICENSE}:=

An event of obtaining a legal license to marry.

MARR {MARRIAGE}:=

Marriage is an official and legal event, defined by the applicable law and customs of the land and the time, that creates a couple, possibly with children. This includes so-called common law marriages.

The name of this record is ill-chosen. The MARR record isn't a marriage record, but a relationship record.
The MARR record can and must be used for all relationship types, with marriage merely being the default relationship type for the couple.

References

- Marriage in GEDCOM
- Same-Sex Marriage in GEDCOM

MARS {MARR_SETTLEMENT}:=
An event of creating an agreement between two people contemplating marriage, at which time they agree to release or modify property rights that would otherwise arise from the marriage.

MEDI {MEDIA}:=
Information about the media or having to do with the medium in which information is stored.

NAME {NAME}:=
Depending on context, a product name, repository name or an individual's full name.
The NAME *must not* contain nobility titles. The NAME *may* contain earned titles and salutations (See NPFX, page 127).
More than one NAME record should be used for individuals known by multiple names.

NATI {NATIONALITY}:=
The nationality of an individual.

NATU {NATURALISATION}:=
The event of obtaining citizenship.

NCHI {NUMBER_OF_CHILDREN}:=
The number of children that this individual (INDI.NCHI) or couple (FAM.NCHI) has.

NICK {NICKNAME}:=
A descriptive or familiar that is used instead of, or in addition to, one's proper name.

NMR {NUMBER_OF_RELATIONSHIPS}:=
The number of relationships (FAM records as a partner) this person would occur in if all relationships were included.

NOTE {NOTE}:=
Additional information provided by the submitter for understanding the enclosing data.

NPFX {NAME_PREFIX}:=
Text which appears on a name line before the given and surname parts of a name.
e.g. **Lt. Cmndr.** Joseph /Allen/ Jr.
In this example Lt. Cmndr. is considered as the name prefix portion.

NSFX {NAME_SUFFIX}:=
Text which appears on a name line after or behind the given and surname parts of a name.
e.g. Lt. Cmndr. Joseph /Allen/ Jr.
In this example Jr. is considered as the name suffix portion.

OBJE {OBJECT}:=
Pertaining to a grouping of attributes used in describing something. Usually referring to the data required to represent a multimedia object, such an audio recording, a photograph of a person, or an image of a document.

OCCU {OCCUPATION}:=
The type of work or profession of an individual.

PAGE {PAGE}:=
A number or description to identify where information can be found in a referenced work.

PEDI {PEDIGREE}:=
Information pertaining to an individual to parent lineage chart.

PHON {PHONE}:=
A unique number assigned to access a specific telephone.

PLAC {PLACE}:=
A jurisdictional name to identify the place or location of an event.

POST {POSTAL_CODE}:=
A code used by a postal service to identify an area to facilitate mail handling.

PROB {PROBATE}:=
An event of judicial determination of the validity of a will. May indicate several related court activities over several dates.

PROP {PROPERTY}:=
Pertaining to possessions such as real estate or other property of interest.

PUBL {PUBLICATION}:=
Refers to when and/or where a work was published or created.

QUAY {QUALITY_OF_DATA}:=
An assessment of the certainty of the evidence to support the conclusion drawn from evidence.

REFN {REFERENCE}:=
A description or number used to identify an item for filing, storage, or other reference purposes.

RELA {RELATIONSHIP}:=
A relationship value between the indicated contexts.

RELI {RELIGION}:=
A religious denomination to which a person is affiliated or for which a record applies.

REPO {REPOSITORY}:=
An institution or person that has the specified item as part of their collection(s).

RESI {RESIDENCE}:=
An address or place of residence that a family or individual resided.

RETI {RETIREMENT}:=
The event of ending one's occupational career.

RIN {REC_ID_NUMBER}:=
A number assigned to a record by an originating automated system that can be used by a receiving system to report results pertaining to that record.

ROLE {ROLE}:=
A name given to a role played by an individual in connection with an event.

ROMN {ROMANISED}:=
A romanised transcription of a superior text string.

SEX {SEX}:=
Indicates the sex of an individual; male, female, intersex or unknown.

SOUR {SOURCE}:=
The initial or original material from which information was obtained or (`HEAD.SOUR`) the system that created the GEDCOM file.

SPFX {SURN_PREFIX}:=
A name piece used as a non-indexing pre-part of a surname.

STAE {STATE}:=
A geographical division of a larger jurisdictional area (country), such as a province or state.

SUBM {SUBMITTER}:=
An individual or organization who contributes genealogical data to a file or transfers it to someone else.

SURN {SURNAME}:=
A family name passed on or used by members of a family.

TEXT {TEXT}:=
The exact wording found in an original source document.

TIME {TIME}:=
A time value in a 24-hour clock format, including hours, minutes, and optional seconds, separated by a colon (:). Fractions of seconds are shown in decimal notation.

TITL {TITLE}:=
A description of a specific writing or other work, such as the title of a book when used in a source context, or a formal designation used by an individual in connection with positions of royalty or other social status, such as Grand Duke.

TYPE {TYPE}:=
A further qualification to the meaning of the superior record. The value does not have any computer processing reliability. It is more in the form of a short one or two word note that should be displayed any time the associated data is displayed.

WIFE {WIFE}:=
A partner in a FAM (family group), often female, often partner to a man, and a biological, official or legal parent to each of the children of this couple.
The name of this record strongly suggests that the line value must identify a wife, but that is not the case; the relationship need not be a marriage, and the individual need not be female, it may identify a man in a gay relationship.

References

Same-Sex Marriage in GEDCOM

WILL {WILL}:=
A legal document treated as an event, by which a person disposes of his or her estate, to take effect after death. The event date is the date the will was signed while the person was alive. (See also PROB (probate), page 128.)

WWW {WEB}:=
World Wide Web address.

Appendix B ANSEL to Unicode Conversion

ANSEL to Unicode Conversion

There is more to conversion from ANSEL (or LANSEL) to Unicode than just the ANSEL to Unicode conversion tables. A major issue is that, in ANSEL, combining characters (modifiers) come before the base characters, while in Unicode they come after the base character.

After conversion from ANSEL to Unicode, the result should be normalised as appropriate for the platform. On Mac OS X, the string must be converted to Unicode Normal Form D (decomposed characters), while on Windows, it must be converted to Unicode Normal Form C (precomposed characters).
This normalisation step must not be skipped for either platform or normal form, as ANSEL text may contain both composed and decomposed characters.

ANSEL to Unicode Conversion Algorithm

1. Convert each ANSEL code point into the corresponding Unicode code point
2. mirror the positions of each base character and its modifiers (modifiers after instead of before base characters)
3. Convert to Unicode Normal Form C or Unicode Normal Form D, as appropriate for the platform

Unicode to ANSEL Conversion

Conversion from Unicode to ANSEL may seem simple; just perform the ANSEL to Unicode conversion step in reverse order:

1. Make sure the Unicode string is in Unicode Normal Form D
2. mirror the positions of each base characters and its modifiers (modifiers before instead of after base characters)
3. Replace Unicode code points with their ANSEL equivalents

However, Unicode to ANSEL conversion is more complicated than that. There are two issues that require attention:

- unsupported characters
- ANSEL's pre-composed characters

Unsupported Characters

First of all, ANSEL does not support all the same characters that Unicode supports, and does not feature an Replacement Character like Unicode does. The widespread convention for such cases is to replace the unsupported character by a question mark. When doing so, care must be taken to replace an unsupported character and all its modifiers by just one single question mark.

ANSEL does not support all the same combining characters as Unicode either. Generally speaking, when a modifier isn't supported, the Unicode character isn't supported, and should be replaced by a question mark.
It is arguably better, albeit inconsistent, to keep the base character if supported, and simply lose any unsupported modifiers.

ANSEL's Pre-Composed Characters

Special attention must be paid to ANSEL's pre-composed characters. For example, ANSEL does not contain an equivalent for *U+031B Combining Horn*, but it does contain equivalents for the pre-composed characters *U+101A Latin Capital Letter O with Horn* and *U+10AF Latin Capital Letter U with Horn*.

If your Unicode to ANSEL conversion fails to take the existence of ANSEL's pre-composed characters into account, your conversion will reduce those characters to their base character.

Unicode to ANSEL Conversion Algorithm

1. Make sure the Unicode string is in Unicode Normal Form D
2. De-normalise for pre-composed characters supported by ANSEL
3. mirror the positions of each base characters and its modifiers (modifiers before instead of after base characters)
4. Replace Unicode code points with their ANSEL equivalents
   - Replace unsupported characters with a question mark
   - Take care to replace an unsupported characters and all its modifiers by a single question mark
   - If a base character is supported, but a modifier is not, keep the base character, but leave off the modifier

The de-normalisation step is essential for taking advantage of ANSEL's pre-composed characters, and not ending up with just their base character instead.

Operating System Support

It is generally unwise to try and roll your own character set conversion functions, and best to rely on built-in functions for any character set handling and conversions. All major operating systems, operating environments, and several third-party libraries provide character set conversion functions, but no major systems and few libraries support ANSEL. However, all major operating systems provide functions for conversion of strings to Unicode Normal Forms:

- Microsoft Windows provides the `NormalizeString()` and `IsNormalizedString()` functions.
- Apple Mac OS X provides the `NSString()` function.
- Oracle Java provides the `Normalizer2` class, with multiple methods.
- Microsoft .NET provides the `String.Normalize()` method.
- Google Android provides the same `Normalizer2` class.
- Apple iOS provides the same `NSString()` function as Apple Mac OS X.

Best Practice for Genealogy Applications

- support import of ANSEL GEDCOM files
- do not export to ANSEL GEDCOM
  - always export to UTF-8 or UTF-16
- use built-in functions for Unicode normalisation

References

- ANSEL Administratively Withdrawn
- GEDCOM 5.5.1 Specification ANSEL Table
- LDS ANSEL versus LDS ANSEL
- ANSEL Alif & Ayn
- ANSEL / Unicode Conversion Tables
- ANSEL / Unicode Conversion Algorithms

Appendix C ANSEL Character Set

ANSEL Tables

ANSEL

The ANSEL character set (ANSI/NISO Z39.47-1985) was originally created in 1985 CE by the Standards Committee on Coded Character Sets for Bibliographic Information Interchange of America. ANSEL is an 8-bit character set that supports dozens of Western languages without requiring multiple, conflicting code pages to do so. ANSEL became widely used in library systems, but has effectively been replaced by Unicode. The ANSEL specification was last updated in 1993 CE, reaffirmed in 2003 CE, and administratively withdrawn in 2013 CE. ANSEL is mostly of historic interest now.

Left Hook

The ANSEL character 0xF7 left hook appears as “left hoof” in the ANSEL standard. That is an acknowledged typographical error. The actual name is “left hook”.

Replaced by Unicode

FamilySearch GEDCOM 4.0 (1989 CE) through 5.5.1 (1999 CE) recommend the use of ANSEL. The *GEDCOM 5.5.1 Annotated Edition* (2018 CE) already notes that the GEDCOM 5.5.1 rules are obsolete, that all genealogy applications should be Unicode-based and always export to Unicode.
GEDCOM 5.5.5 recommends that GEDCOM writers use Unicode instead of ANSEL.

Being phased out

Use of ANSEL in GEDCOM files is being phased out.
Use of the ANSEL character set is officially deprecated now.

New versions of existing systems *need not* and *should* provide the ability to export ANSEL-encoded GEDCOM file, but should be able to import ANSEL GEDCOM files, particularly ANSEL GEDCOM files created by a previous version of the same system.
New systems using GEDCOM 5.5.5 *must not* export ANSEL-encoded GEDCOM files, and *need not* support import of any ANSEL GEDCOM files.

LANSEL

Not ANSEL, but LANSEL

Technically, the tables provided in the GEDCOM specification aren't ANSEL tables, but LANSEL tables.
The FamilySearch tables leave ANSEL characters out and add extensions. Because these extensions are defined by the LDS (FamilySearch's parent company), they are known as LDS Extensions, and the resulting character set is known as LDS ANSEL, LANSEL for short.
GEDCOM keeps using the value `ANSEL` to specify the LANSEL character set for backward compatibility's sake.

Bad Tables

The LANSEL tables in GEDCOM 5.5.1 and earlier often do not display as intended, because FamilySearch relied on WordPerfect fonts that most users do not have installed. The *GEDCOM 5.5.1 Annotated Edition* contains LANSEL tables that does not rely on any particular font.

Multiple Versions

Different versions of FamilySearch GEDCOM contain different “ANSEL” tables. Different FamilySearch GEDCOM versions omit different ANSEL characters and add different extensions. Thus, FamilySearch actually defined different versions of LANSEL in different GEDCOM versions.
That creates unnecessary uncertainty about whether specific characters are supported or not.
The *GEDCOM 5.5.1 Annotated Edition* provides consolidated LANSEL tables as bonus information.

Consolidated Tables

The consolidated tables are a combination of the original tables in the ANSEL standard and the LANSEL tables that FamilySearch published in different GEDCOM versions. The consolidated tables contain all the ANSEL characters that FamilySearch omitted, as well as all the LDS extensions they added in different GEDCOM versions.

Recommended Tables

GEDCOM 5.5.5 provides the same consolidated LANSEL table as *GEDCOM 5.5.1 Annotated Edition* as the replacement for the tables provided in earlier versions of GEDCOM. Developers are recommended to not rely on those earlier tables, but simply use the consolidated tables provided here with previous versions of GEDCOM as well.

WordPerfect Codes

The LANSEL tables in GEDCOM 5.5.1 and earlier contain a column specifying wpcodes; the WordPerfect code page and code point for a character. That irrelevant information is no longer provided.

Unicode

The new LANSEL tables provide the Unicode code points and names that correspond with the LANSEL codes and the Unicode names (technically, the Unicode names should be ALL-UPPERCASE, but that does not help readability). It is recommended that developers use the Unicode names exclusively.

There is more to conversion from ANSEL or LANSEL to Unicode than just the ANSEL to Unicode conversion tables. See Appendix B *ANSEL to Unicode Conversion*, page 130.

Combining Characters

The graphics column of these tables, show the *actual* combining characters, not some non-combining character that approximates it. For example, the first character shown is U+0300 Combining Grave Accent, not U+0060 Grave Accent.
To show the combining characters correctly and to clearly show the relative position of the combining character, the table uses ◌ (U+25CC Dotted Circle) as a placeholder character.

ANSEL Non-spacing graphic characters

Hex	Dec	Graphic	Name	example of use	code point	Name
E0	224	◌̉	low rising tone mark	củi	U+0309	Combining Hook Above
E1	225	◌̀	grave accent	règle	U+0300	Combining Grave Accent
E2	226	◌́	acute accent	está	U+0301	Combining Acute Accent
E3	227	◌̂	circumflex accent	même	U+0302	Combining Circumflex Accent
E4	228	◌̃	tilde	niño	U+0303	Combining Tilde
E5	229	◌̄	macron	gājējs	U+0304	Combining Macron
E6	230	◌̆	breve	altă	U+0306	Combining Breve
E7	231	◌̇	dot above	żaba	U+0307	Combining Dot Above
E8	232	◌̈	umlaut (diaeresis)	öppna	U+0308	Combining Diaeresis
E9	233	◌̌	hacek	vždy	U+030C	Combining Caron
EA	234	◌̊	circle above (angstrom)	hår	U+030A	Combining Ring Above
EB	235	◌͡	ligature, left half	akademiia͡	U+FE20	Combining Ligature Left Half
EC	236	◌͜	ligature, right half	akademiia͜	U+FE21	Combining Ligature Right Half
ED	237	◌̓	high comma, off center	rozdel̓ovac	U+0315	Combining Comma Above Right
EE	238	◌̋	double acute accent	időszaki	U+030B	Combining Double Acute Accent
EF	239	◌̑	candrabindu	Aliĭev	U+0310	Combining Cadrabindu
F0	240	◌̧	cedilla	ça	U+0327	Combining Cedilla
F1	241	◌̨	right hook, ogonek	vietą	U+0328	Combining Ogonek
F2	242	◌̣	dot below	teḍa	U+0323	Combining Dot Below
F3	243	◌̤	double dot below	khuṯbah	U+0324	Combining Diaeresis Below
F4	244	◌̥	circle below	Samskṛta	U+0325	Combining Ring Below
F5	245	◌̳	double underscore	G̳hulam	U+0333	Combining Double Low Line
F6	246	◌̲	underscore	s̲amar	U+0332	Combining Low Line
F7	247	◌̦	left hook	dārziņa	U+0326	Combining Comma Below
F8	248	◌̡	right cedilla	khọng	U+031C	Combining Left Half Ring Below
F9	249	◌̮	half circle below (upadhmaniya)	ḫumantuš	U+032E	Combining Breve Below
FA	250	◌͠	double tilde, left half	n͠galan	U+FE22	Combining Double Tilde Left Half
FB	251	◌͡	double tilde, right half	n͡galan	U+FE23	Combining Double Tilde Right Half
FC	252	◌̸	diacritic slash through char		U+0338	Combining Long Solidus Overlay
FD	253	◆	*unused*		U+FFFD	Replacement Character
FE	254	◌̔	high comma, centered	g̔eotermika	U+0313	Combining Comma Above
FF	255	◆	*illegal*		U+FFFD	Replacement Character

ANSEL Spacing graphic characters

Hex	Dec	Graphic	Name	example of use	code point	Name
A0	160		*unused*		U+FFFD	Replacement Character
A1	161	Ł	slash L — uppercase	Łódź	U+0141	Latin Capital Letter L with Stroke
A2	162	Ø	slash O — uppercase	Øst	U+00D8	Latin Capital Letter O with Stroke
A3	163	Đ	slash D — uppercase	Đuro	U+0110	Latin Capital Letter D with Stroke
A4	164	Þ	thorn — uppercase	Þann	U+00DE	Latin Capital Letter Thorn
A5	165	Æ	ligature AE — uppercase	Ægir	U+00C6	Latin Capital Letter AE
A6	166	Œ	ligature OE — uppercase	Œuvre	U+0152	Latin Capital Ligature OE
A7	167	◌′	mjagkij znak	fakul′tet	U+02B9	Modifier Letter Prime
A8	168	·	middle dot	novel·la	U+00B7	Middle Dot
A9	169	♭	musical flat	B♭	U+266D	Musical Flat Sign
AA	170	®	registered trademark	ABC®	U+00AE	Registered Sign
AB	171	±	plus or minus	A±B	U+00B1	Plus-Minus Sign
AC	172	Ơ	hook O - uppercase	BƠ	U+01A0	Latin Capital Letter O with Horn
AD	173	Ư	hook U - uppercase	XƯA	U+01AF	Latin Capital Letter U with Horn
AE	174	◌’	alif	Un’yusho	U+02BC	Modifier Letter Apostrophe
AF	175		*unused*		U+FFFD	Replacement Character
B0	176	◌‘	ayn	fa‘il	U+02BB	Modifier Letter Turned Comma
B1	177	ł	slash l— lowercase	rozbił	U+0142	Latin Small Letter L with Stroke
B2	178	ø	slash o— lowercase	høj	U+00F8	Latin Small Letter O with Stroke
B3	179	đ	slash d— lowercase	đavola	U+0111	Latin Small Letter D with Stroke
B4	180	þ	thorn— lowercase	þann	U+00FE	Latin Small Letter Thorn
B5	181	æ	ligature ae— lowercase	skæg	U+00E6	Latin Small Letter AE
B6	182	œ	ligature oe— lowercase	œuvre	U+0153	Latin Small Ligature OE
B7	183	◌″	hard sign (tvjordyj znak)	ob″i͡avlenie	U+02BA	Modified Letter Double Prime
B8	184	ı	dotless i— lowercase	masalı	U+0131	Latin Small Letter Dotless I
B9	185	£	British pound	£5.00	U+00A3	Pound Sign
BA	186	ð	eth	verður	U+00F0	Latin Small Letter Eth
BB	187		*unused*		U+FFFD	Replacement Character
BC	188	ơ	hook o - lowercase	Sơ	U+01A1	Latin Small O with Horn
BD	189	ư	hook u - uppercase	Tự Đức	U+01B0	Latin Small U with Horn
BE	190	□	empty box (LDS-extension)		U+25A1	Empty Box
BF	191	■	black box (LDS-extension)		U+25A0	Black Box
C0	192	°	degree sign	10°C.	U+00B0	Degree Sign
C1	193	ℓ	script l	25 ℓ.	U+2113	Script Small L
C2	194	℗	phono copyright mark	Decca℗	U+2117	Sound recording copyright
C3	195	©	copyright mark	©1993	U+00A9	Copyright Sign
C4	196	♯	music sharp sign	D♯	U+266F	Music Sharp Sign
C5	197	¿	inverted question mark	¿Qué?	U+00BF	Inverted Question Mark
C6	198	¡	inverted exclamation mark	¡Esta!	U+00A1	Inverted Exclamation Mark
C7	199		*unused*		U+FFFD	Replacement Character
C8	200		*unused*		U+FFFD	Replacement Character
C9	201		*unused*		U+FFFD	Replacement Character
CA	202		*unused*		U+FFFD	Replacement Character
CB	203		*unused*		U+FFFD	Replacement Character
CC	204		*unused*		U+FFFD	Replacement Character
CD	205	e	e in middle of line (LDS extension)		U+0065	Latin Small Letter E
CE	206	o	o in middle of line (LDS extension)		U+006F	Latin Small Letter O
CF	207	ß	Ess Zed	Preußen	U+00DF	Latin Small Letter Sharp S

End of Specification

The end of the GEDCOM 5.5.5 specification.

Bonus Chapters

The Bonus Chapters are not part of the GEDCOM Specification proper.

GEDCOM Validation

Developers must make sure that the GEDCOM files exported by their application are valid GEDCOM files. There are several GEDCOM validators and some other tools available to help with that. With the exception of GedChk, these tools are fairly mature.

GEDCOM validators are themselves applications that may contain errors.
When in doubt, try multiple validation tools, and re-read the specification.

GEDCOM Validators

GedChk

GedChk is a free GEDCOM validator by FamilySearch. GedChk is a command-line application for MS-DOS, for which only version 0.9 Beta is available. GedChk supports GEDCOM 5.5 and 5.5.1. GedChk is available for FTP from the LDS website. GedChk is not well-behaved enough to run in a Windows DOS Box, but it does run in DOSBox, which is freely available for many operating systems.

VGedX

VGedX is a free GEDCOM validator created by Tim Forsythe, the creator of Gigatrees.
VGedX is a command-line application for Windows, complemented by a windowed configuration interface. VGedX supports GEDCOM 5.5, GEDCOM 5.5.1 and GEDCOM 5.6, and the user interface is available in multiple languages. VGedX is available for Windows, and requires 64-bit Windows 7 or later.

Chronoplex GEDCOM Validator

The Chronoplex GEDCOM Validator is a free product of Chronoplex (Andrew Hoyle), creators of Chronoplex Family Tree. The Chronoplex GEDCOM Validator is a Microsoft .NET application for Windows. The Chronoplex GEDCOM Validator supports GEDCOM 5.5, GEDCOM 5.5.1 and GEDCOM 5.6, and the user interface is available in multiple languages. The Chronoplex GEDCOM Validator requires Windows and Microsoft .NET.

GED-inline

GED-inline is a free on-line GEDCOM validator by Nigel Munro Parker.
GED-inline supports GEDCOM 5.5 and GEDCOM 5.5.1.

Additional Validation Tools

Behold

Behold is a commercial genealogy viewer by Louis Kessler. Behold's forgiving GEDCOM reader provides so many error and warning messages, that Behold be thought of as a GEDCOM validator in disguise.
Behold supports every version of GEDCOM, including GEDCOM 1.0. Behold requires Windows NT 4.0 or later.

GedPad

GedPad is free product of Nigel Button Software, makers of The Complete Genealogy Reporter. GedPad isn't a validator, but a GEDCOM editor with some validation features. GedPad's user-interface is a bit unusual and it does not support ASCII or ANSEL files, but it can be a handy tool

to have around when some GEDCOM file gives you trouble.

Genealogica Grafica

Genealogica Grafica is a free product by Tom de Neef, and the successor to his previous product, KStableau. Genealogica Grafica is a tool for creation of web charts and reports. On loading a GEDCOM file, it provides a report that includes GEDCOM errors and genealogy consistency checks. Genealogica Grafica support GEDCOM 5.5 and GEDCOM 5.5.1. Genealogica Grafica requires Windows.

References

- GEDCOM Validation
- DOSBox
- ftp://ftp.ldschurch.org/genealogy/GEDCOM/Utility/GedChk/
- VGedX, the Gigatrees GEDCOM Validator
- Chronoplex GEDCOM Validator
- GED-inline
- Behold
- GedPad
- Genealogica Grafica

GEDCOM Version Detection

Examine First

A GEDCOM reader cannot simply start reading a GEDCOM file. The various GEDCOM versions, character encodings and line terminators possible create a catch-22. A GEDCOM reader must start by *examining* the GEDCOM header, to discover the GEDCOM version, character encoding and line terminator used, before it can start *reading* the GEDCOM file - including the GEDCOM header. The GEDCOM reader can and should detect errors in the GEDCOM header.

Systems supporting GEDCOM need two separate things to read GEDCOM files; a GEDCOM header recognition routine and at least one GEDCOM reader. A typical system will have more than one GEDCOM reader, in support of multiple GEDCOM versions. The different readers take advantage of many of the same subroutines, such as those for character set conversion, but do parse GEDCOM files differently.

Choosing a GEDCOM Reader

The system chooses which GEDCOM reader to use for a particular GEDCOM file after examining the GEDCOM header and interpreting the results. A system should only try to read a particular GEDCOM file if it has a GEDCOM reader that supports its GEDCOM version and encoding. If a system does not support it, the system should admit implementation limitation, with a message such as “GEDCOM 4.0 is not supported. Please use GEDCOM 5.5 or later.”, and abort processing.

When the system picks a GEDCOM reader, it has not checked that the GEDCOM header is valid. It has merely examined the header to pick the right GEDCOM reader. It is up to the chosen GEDCOM reader to read the entire GEDCOM header and verify that the GEDCOM header is valid.

GEDCOM 5.5 & 5.5.1 versus GEDCOM 5.5.5

Many existing systems have a single GEDCOM reader that supports both GEDCOM 5.5 and GEDCOM 5.5.1. Many of these readers accept errors known to occur in third-party GEDCOM files, such as illegal tags and invalid GEDCOM headers. After more than two decades of systems producing sloppy GEDCOM 5.5 and 5.5.1 files, many GEDCOM 5.5.x readers are quite burdened by the complexity of supporting many known third-party errors.
Those GEDCOM 5.5 and 5.5.1 errors do not occur in GEDCOM 5.5.5 files, so a GEDCOM 5.5.5 reader need not and should not be burdened with that complexity.
Besides, GEDCOM 5.5.5 readers are not allowed to accept invalid GEDCOM.

Genealogy software developers *must not* extend their existing GEDCOM 5.5.1 reader to accept GEDCOM 5.5.5 files. Developers *must* create a new, separate GEDCOM 5.5.5 reader. GEDCOM 5.5.5 is a leaner, meaner GEDCOM, less complex and simpler to parse. This new reader should have much simpler and cleaner code, and be noticeable faster than the old GEDCOM 5.5.1 reader. The GEDCOM 5.5.5 reader *must* make sure the GEDCOM header is valid, and reject the ostensible GEDCOM file if it is not.

Newer GEDCOM Versions

If the HEAD.GEDC.VERS value is a higher value that the system does not recognise yet, the system *must* admit this implementation limitation, and abort. The system *must not* try to import the GEDCOM file until explicit support for the new GEDCOM version has been added.

GEDCOM 5.5.5

A GEDCOM 5.5.5 writer *must* create a correct GEDCOM header, and a GEDCOM reader *must not* accept invalid GEDCOM headers. Detection of GEDCOM 5.5.5 files is quite straightforward,

especially in comparison to the complexity of correctly distinguishing between GEDCOM 5.5 and 5.5.1 files. A system will call upon its GEDCOM 5.5.5 reader after detecting a HEAD.GEDC.VERS value of `5.5.5`.
When called upon, the GEDCOM 5.5.5 reader starts reading the GEDCOM file from the beginning, starting with the GEDCOM header. If the header is not completely valid, the file *must* be rejected as *not GEDCOM*, specifically as not even having a correct GEDCOM header.

GEDCOM 5.5.1

Many GEDCOM readers for GEDCOM 5.5.1 and earlier are quite forgiving about GEDCOM errors including errors in the GEDCOM header. This is a very undesirable bad practice, that effectively licenses the creators of sloppy GEDCOM with bad GEDCOM headers to continue doing so. The existing GEDCOM readers for GEDCOM 5.5.1 and earlier should continue their current practices for backward compatibility's sake, so as to not break existing user practices.

All developers are encouraged to fix problems with the GEDCOM files produced by their existing systems. That starts by making sure the system produces a correct and valid GEDCOM header.
Developers are not merely encouraged to take advantage of GEDCOM validators to verify the GEDCOM produced by their systems, but strongly encouraged to make validator usage an integral part of their software development cycle.

Developers of systems that do not support GEDCOM 5.5.5 yet, should make sure that their systems recognise GEDCOM 5.5.5, admits the implementation limitation with a message such as “GEDCOM 5.5.5 not supported yet”, and then aborts processing.

Developers are encouraged to add GEDCOM 5.5.5 support as soon as possible.

GEDCOM 5.5.1 Version Detection

GEDCOM 5.5.1 version detection is complicated by the fact that several applications produce GEDCOM 5.5.1 files that incorrectly claim to be version `5.5` files instead of `5.5.1` files. In fact, nowadays (2019 CE), most GEDCOM files that claim to be GEDCOM 5.5 files are actually GEDCOM 5.5.1 files.

FamilySearch PAF is the best known application to lie about the GEDCOM version used, but not the only one. Other products that use GEDCOM 5.5.1 but lie that they are using GEDCOM 5.5 include Ancestral Quest, MyHeritage Family Tree Builder, Ancestry.com's New Family Tree Maker, Family Tree Heritage, Geni.com, GEDitCOM II, gramps, Heredis 98, Kith and Kin, Legacy Family Tree, Ancestry.com's Online Family Tree, Personal Ancestry Writer II, Reunion, The Next Generation of Genealogy Sitebuilding (TNG), and WikiTree.

Some products that correctly identify their use of GEDCOM 5.5.1 include Ahnenblatt, Family Echo, Ancestry.com's New Family Tree Maker, findmypast Family Tree, gedcom4j, Legacy Family Tree, MacFamilyTree, Picturae's Memorix Maior, phpGedView, RootsMagic, and WebTrees.

Some products appear in both list because some versions of the product get it wrong, while later versions get it right. For the detection algorithm presented here, all that matters is when products started using GEDCOM 5.5.1 yet continued to label their GEDCOM files (incorrectly) as GEDCOM 5.5. It does not matter whether or when the developers fixed their product to start labelling their GEDCOM 5.5.1 files correctly.

GEDCOM 5.5 versus GEDCOM 5.5.1 File Content

It is possible to distinguish between GEDCOM 5.5 and GEDCOM 5.5.1 files based on their content, for example with the following rules:

- if the GEDCOM files contains a OBJE.BLOB record, it is GEDCOM 5.5
- if the GEDCOM files contains a PLAC.MAP record, it is GEDCOM 5.5.1

However, a GEDCOM reader should be able to determine the actual GEDCOM version without looking beyond the GEDCOM header.

GEDCOM 5.5.1 Detection Algorithm

This is a the algorithm for distinguishing between GEDCOM 5.5 and GEDCOM 5.5.1. Numbered steps should be performed in the order shown. As soon as a rule matches and the actual GEDCOM version is known, no further tests should be performed.

- Detect the character encoding first
- read the HEAD.GEDC.VERS line value
- if the value is `5.5.1`, it is GEDCOM 5.5.1
- if the value is `5.5`, it may be either GEDCOM 5.5 or GEDCOM 5.5.1

    1. (Optional): check the system identifier (HEAD.SOUR line value)
       against a list of system identifiers for products discontinued before (or some time after) the introduction of GEDCOM 5.5.1:
       if there is a match, it's definitely GEDCOM 5.5, and not GEDCOM 5.5.1.

    2. Check the GEDCOM encoding
        - if the character encoding is UTF-8, it is GEDCOM 5.5.1

    3. Check for tags introduced in GEDCOM 5.5.1
        - if the header contains a HEAD.SOUR.CORP.ADR3 record, it is GEDCOM 5.5.1
        - if the header contains a HEAD.SOUR.CORP.EMAIL record, it is GEDCOM 5.5.1

- if the header contains a HEAD.SOUR.CORP.FAX record, it is GEDCOM 5.5.1
- if the header contains a HEAD.SOUR.CORP.WWW record, it is GEDCOM 5.5.1

4. Check for known products and versions.

    1. Make sure you have a system identifier.
        - if the system identifier is too long (it is for The Next Generation of Genealogy Sitebuilding), issue a non-fatal error and continue
        - If the (HEAD.SOUR line value is missing (illegal!)
            - try to identify the correct system identifier by other means (see section on MacFamilyTree below)
            - if you failed to identify it, issue a fatal error and abort
            - if you managed to identify it, issue a non-fatal error and continue with that system identifier

    2. Make sure you have a version number.
        - If the (HEAD.VERS line value is missing, issue a fatal error and abort.
        - If the (HEAD.VERS line value isn't a proper version number
            - try to extract the version number (see section on Family Tree Maker below)
            - if you failed to extract it, issue a fatal error and abort
            - if you managed extract it, issue a non-fatal error and continue with that version number

    3. Perform the known product / version number checks.
        - if the system identifier is `PAF`
            - if the version number is less than `5.0`, it is GEDCOM 5.5
            - if the version number is `5.0` or more, it is GEDCOM 5.5.1
        - See the table below for more products and version numbers.

5. Check for user-defined tags being used instead of GEDCOM 5.5.1 tags
    - if the header contains a HEAD.SOUR.CORP._EMAIL record, it must be GEDCOM 5.5
    - if the header contains a HEAD.SOUR.CORP._FAX record, it must be GEDCOM 5.5
    - if the header contains a HEAD.SOUR.CORP._WWW record, it must be GEDCOM 5.5

6. None of the previous rules matched? Assume it is 5.5 as stated.

- if the HEAD.GEDC.VERS value is `5.5`, but the actual GEDCOM version is 5.5.1, issue a non-fatal error
- Issue an informative message that the file will be processed as GEDCOM 5.5 or GEDCOM 5.5.1, whichever version was detected

Notice the last two instructions in the algorithm. It is good to let the user know that an ostensible GEDCOM 5.5 file is really a GEDCOM 5.5.1 file. It is good to explicitly inform the user how the file will be processed.

Product Table

Version of product that started using GEDCOM 5.5.1		
application	**system identifier**	**version**
Personal Ancestral File	PAF	5.0
Ancestral Quest	AncestQuest	12.0
Family Tree Maker	FTM	21.0.0.466
GenoPro	GenoPro	2.0
gramps	Gramps	2.0
Lifelines	Lifelines	3.0
MacFamilyTree	MacFamilyTree	5.7.8
MagiKey Family Tree	MagiKey Family Tree	*all versions*
Family Tree Builder	MYHERITAGE	5.5
Reunion	Reunion	9.0
RootsMagic	RootsMagic	*all versions*
The Next Generation of Genealogy Sitebuilding	The Next Generation of Genealogy Sitebuilding	7.0
PRO-GEN	PRO-GEN	3.0
This list is not exhaustive.		

Product Information

Family Tree Maker Version Number

The Family Tree Maker name is used for what are really three different, consecutive products (and then there's also Family Tree Maker for Macintosh):

- the original Family Tree Maker, an MS-DOS application
- Family Tree Maker for Windows a Windows applications
- New Family Tree Maker for Windows, a Microsoft .NET application

GEDCOM System Identifiers

Those three different products use only two different GEDCOM system identifiers. Family Tree Maker for MS-DOS uses `FTM` Family Tree Maker for Windows uses `FTW` and New Family Tree Maker for Windows uses `FTM` again.
This is wrong, and may mess up some tests, but does not mess up this test, because the version numbers continued to increase with new products. Family Tree Maker 2012 Service Pack 2 (version 21.0.0.46) is the first version to produce ostensible GEDCOM 5.5 files that are really GEDCOM 5.5.1 files. Family Tree Maker for MS-DOS pre-dates GEDCOM 5.5.1, and all its versions numbers are considerably lower than those of New Family Tree Maker for Windows. So, if the Family Tree Make version number is 21.0.0.466 or later, it is GEDCOM 5.5.1.

FTM Version Number

When Ancestry.com created New Family Tree Maker for Windows, they messed up the version number record, by practically treating it as product name record. This was only fixed after Software MacKiev became the owner of Family Tree Maker.
Family Tree Maker 2012's version number is `21.0.0.38`, but the HEAD.SOUR.VERS line value isn't `21.0.0.38`, it is `Family Tree Maker (21.0.0.38)` instead. That value is illegal, for two reasons:

- it clearly isn't a proper version number
- this value (29 characters) is longer than the maximum HEAD.SOUR.VERS line value length

of 15 code units

A GEDCOM reader should definitely issue an error when encountering this, and it is okay for that error to be an fatal error; it is fine to reject an ostensible GEDCOM file if it does not even have a valid GEDCOM header.
It is not difficult to recognise the `Family Tree Maker (`*`version number`*`)` pattern and extract the version number from it, and this is what a GEDCOM reader has to do before it can perform this test.

32-bit versus 64-bit Version Number

From Family Tree Maker 2014 onwards, Family Tree Maker comes in both a 32-bit and a 64-bit build, and those two different builds have different version numbers. The version number of the 32-bit Family Tree Maker 2014 is `22.0.0.207`, while the version number of the 64-bit Family Tree Maker 2014 is `22.0.0.1207`.
This does not complicate the test, as the switch to GEDCOM 5.5.1 happened with version Family Tree Maker 2012 Service Pack 2 (version 21.0.0.46), for which there is only a 32-bit build, and therefore just one version number to compare to.

Family Tree Maker for Macintosh

There are two Family Tree Maker for Mac products.

- Family Tree Maker for Mac, based on Family Tree Maker for Windows, released by Broderbund in 1997 CE
- Family Tree Maker for Mac, based on New Family Tree Maker, released by Ancestry.com in 2010

The new Family Tree Maker for Mac uses the same system identifier (`FTM`) as New Family Tree Maker (for Windows), and that is wrong, because it is definitely another product. Family Tree Maker for Mac uses the same version numbering as New Family Tree Maker, and the older Ancestry.com Family Tree Maker for Mac also uses the same illegal HEAD.SOUR.VERS format. The GEDCOM 5.5.1 detection code for Family Tree Maker does not to distinguish between Family Tree Maker (for Windows) and Family Tree Maker for Mac.

Recognising MacFamilyTree GEDCOM files

Recognising MacFamilyTree GEDCOM files is harder than it should be. In many MacFamilyTree GEDCOM files, the HEAD.SOUR value is empty. This is illegal, and a GEDCOM reader should definitely issue an error, and it is okay for that error to be an fatal error, because it's fine to reject an ostensible GEDCOM file if it does not even have a valid GEDCOM header.
It is possible for a GEDCOM reader to recognise the GEDCOM file as MacFamilyTree GEDCOM file; if the HEAD.SOUR line value is empty, but the HEAD.SOUR.NAME line value is `MacFamilyTree`, it is a MacFamilyTree GEDCOM file.

Legacy Family Tree

Legacy Family Tree version 3 and 4 GEDCOM files lack the mandatory GEDCOM version number.
Legacy Family Tree version 3 and 4 use GEDCOM 5.5.

Table Updates

This algorithm can be improved by updating the table with more product-specific information. This bonus chapter is based on and practically identical to the *GEDCOM 5.5.1 Version Detection* article, which is likely to be updated sooner and more frequently than this chapter.

References

- GEDCOM 5.5.1 Version Detection
- GEDCOM Version Detection
- Truncated GEDCOM Version
- GEDCOM System Identifiers
- New Family Tree Maker versions
- Ancestry.com & Software MacKiev Family Tree Maker GEDCOM Header

Event GEDCOM

Back in 1994, CommSoft, the creators of the Roots series of genealogy applications, defined an alternative GEDCOM form. It was originally called InterGED, but best known as Event GEDCOM. Event GEDCOM was first supported by ROOTS IV.
Although Event GEDCOM is arguably superior to the Lineage-Linked Form, Event GEDCOM never caught on. Event GEDCOM was only ever supported by Roots IV, Roots V, Family Gathering and Ultimate Family Tree.
Event GEDCOM files are rare, and backward compatible with GEDCOM 5.3 lineage-linked GEDCOM files.

FamilySearch Misinformation

FamilySearch's statement about CommSoft's Event GEDCOM in GEDCOM 5.4, 5.5, 5.5.1 (and even the 5.6 draft) is incorrect. Event GEDCOM *does not* use the line value "`EVENT_LINEAGE_LINKED`", neither with nor without quotes. Event GEDCOM files are backward compatible with lineage-linked GEDCOM files, and therefore use the line value `LINEAGE-LINKED`, *as instructed by the FamilySearch GEDCOM specification*: “Systems will use this value to specify GEDCOM compatible with these specifications.”.

Actual Event GEDCOM

Event GEDCOM 1.0 files contain a additional subrecord, HEAD.FORM.FORM record with the value `EVENT` (illegal in LINEAGE-LINKED GEDCOM), and, although based on GEDCOM 5.3, claims to be GEDCOM version 1.0.

```
0 HEAD
...
1 SOUR FAMGATH
2 VERS 1.0
2 NAME Family Gathering
2 CORP Palladium Interactive, Inc.
...
1 GEDC
2 VERS 1.0
2 FORM LINEAGE-LINKED
3 FORM EVENT
1 DEST FAMGATH
...
```

Family Gathering 1.0 Event GEDCOM header.

References

- GEDCOM Form
- Event GEDCOM Detection

Known GEDCOM Form Errors

Many GEDCOM 5.5 and 5.5.1 readers correct for known GEDCOM 5.5 and 5.5.1 GEDCOM `LINEAGE-LINKED` spelling errors. A GEDCOM 5.5.5 reader *need not* and *must not* correct for spelling errors.

If a GEDCOM 5.5.5 reader encounters an unknown GEDCOM form, it *must not* continue processing, but must abort with an *unsupported GEDCOM form* message.

`LINEAGE_LINKED` instead of `LINEAGE-LINKED` (underscore instead of dash)

Millennia's Legacy Family Tree (now owned by MyHeritage) is perhaps the best known but not the only genealogy application to get this wrong. This error occurs in GEDCOM files produced by EasyTree, FamilyMatters, GEDitCOM, Généamania, GenoPro, Secere Family Tree and WebSSG.

`LINAGE-LINKED` instead of `LINEAGE-LINKED` (first E missing):

This error occurs in GEDCOM files created by GenoPro.

`Lineage - Linked` instead of `LINEAGE-LINKED` (spaces before and after the dash):

The French application Genealogic by Infoduc and the French site Genealogie.com created GEDCOM files containing this error.

More than One <<SUBMITTER_RECORD>>

A GEDCOM 5.5.5 file contains exactly one <<SUBMITTER_RECORD>>, directly after the <<HEADER>>

A typical GEDCOM 5.5.1 file contains exactly one <<SUBMITTER_RECORD>>, referenced by HEAD.SUBM, but GEDCOM 5.5.1 allowed multiple submitter records.

Both the <<INDIVIDUAL_RECORD>> and the <<FAM_GROUP_RECORD>> optionally reference a submitter record. This allows multiple submitters to be associated with different records.

This GEDCOM 5.5.1 feature remained widely unsupported. It makes little sense for single-user applications, and even collaborative applications (social genealogy sites) do not use it. All genealogy applications, including FamilySearch PAF support only one submitter per GEDCOM file.
The editor and reviewers are not aware of any genealogy application that supports more than submitter per GEDCOM file. Even if there is one, other genealogy applications are not able to import and process that data. The GEDCOM 5.5.1 Annotated Edition strongly deprecated this feature. The GEDCOM 5.5.5 specification demands exactly one submitter record.

GEDCOM Writer Best Practice

- use GEDCOM 5.5.5 or later

GEDCOM 5.5.1 Writer Best Practice

- create exactly one submitter record
- have HEAD.SUBM reference this submitter record
- write the submitter record immediately after the GEDCOM header

GEDCOM 5.5.1 Reader Best Practice

- Remember the HEAD.SUBM cross-reference identifier
- Keep track of submitter record identifiers like you keep track of other identifiers
  - if any cross-reference identifier is reused, issue a fatal error (identifier already used) and abort the import
  - if the referenced submitter record is not found, issue a fatal error (referenced record not found) and abort the import

- Until you have the right submitter record:
  - Accept each submitter record you encounter (overwriting the previous one, if any)
  - issue an error if it isn't the one referenced by HEAD.SUBM

- Once you have the right submitter record, for each additional submitter record:
  - do *not* overwrite the submitter record you already found
  - issue an *implementation limitation* error (only one submitter record supported) and move on

No Standard for Multimedia File Transfer

The GEDCOM specification defines a <<MULTIMEDIA_RECORD>>. Each multimedia record provides a link to an external file. The GEDCOM specification does not provide a standard for bundling multimedia files with a GEDCOM file.

As long as the source and destination system are two different applications on the same computer (or mirrored systems), the multimedia file links are efficient, but for sharing multimedia files to another system, a locally valid link is inadequate.
Links to web resources do remain valid between different systems, but their permanence cannot be guaranteed, and performance is likely to suffer.

The FamilySearch GEDCOM specification does not provide any standard, any rules or guidelines for bundling multimedia files with a GEDCOM file. It is not uncommon to bundle a GEDCOM file and media together in a ZIP file, but that only takes care of the media transfer, not the file paths, which are almost sure to be different on different systems, especially if these are managed by different users.

Some desktop applications have built-in smarts to try and repair files references, and several desktop applications provide syncing of desktop databases with web databases.

Best Practices

Applications should maintain links to media, *not* store media inside their database. Storing media inside the database bloats the database, and makes the media inaccessible to other tools.
Genealogy software developers and applications should encourage users to collect and keep all related media in one directory (with subdirectories).

GEDCOM 5.5 <<MULTIMEDIA_RECORD>>

The FamilySearch GEDCOM 5.5.1 <<MULTIMEDIA_RECORD>> is different from the GEDCOM 5.5 <<MULTIMEDIA_RECORD>>.

This is the GEDCOM 5.5.1 <<MULTIMEDIA_RECORD>> (corrected to use <MULTIMEDIA_FILE_REFERENCE> instead of the non-existing <MULTIMEDIA_FILE_REFN>):

MULTIMEDIA_RECORD:=

n @<XREF:OBJE>@ **OBJE**	{1:1}
+1 **FILE** <MULTIMEDIA_FILE_REFERENCE>	{1:1}
+2 **FORM** <MULTIMEDIA_FORMAT>	{1:1}
+3 TYPE <SOURCE_MEDIA_TYPE>	{0:1}
+2 TITL <DESCRIPTIVE_TITLE>	{0:1}
+1 REFN <USER_REFERENCE_NUMBER>	{0:M}
+2 TYPE <USER_REFERENCE_TYPE>	{0:1}
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}
+1 <<NOTE_STRUCTURE>>	{0:M}
+1 <<SOURCE_CITATION>>	{0:M}
+1 <<CHANGE_DATE>>	{0:1}

The BLOB context of the multimedia record was removed in version 5.5.1. A reference to a multimedia file was added to the record structure. The file reference occurs one to many times so that multiple files can be grouped together, each pertaining to the same context. For example, if you wanted to associate a sound clip and a photo, you would reference each multimedia file and indicate the format using the FORM tag subordinate to each file reference.

The <<SOURCE_CITATION>> line is not new in GEDCOM 5.5.1.
It was added to GEDCOM 5.5 through the GEDCOM 5.5 errata sheet.

The GEDCOM 5.5.1 <<MULTIMEDIA_RECORD>> allows a single OBJE record to link to multiple related files.
This feature is not widely supported; many genealogy application will read only the first OBJE.FILE. To avoid loss of data, it is recommended to have just one OBJE.FILE subrecord per OBJE record. The GEDCOM 5.5.5 specification simplified the multimedia record: one multimedia record per file, one file per multimedia record.

This is the GEDCOM 5.5 <<MULTIMEDIA_RECORD>> for comparison:

MULTIMEDIA_RECORD:=

n @<XREF:OBJE>@ **OBJE**	{1:1}
+1 FORM <MULTIMEDIA_FORMAT>	{1:1}
+1 TITL <DESCRIPTIVE_TITLE>	{0:1}
+1 <<NOTE_STRUCTURE>>	{0:M}
+1 <<SOURCE_CITATION>>	
+1 BLOB	
+2 CONT <ENCODED_MULTIMEDIA_LINE>	
+1 OBJE @<XREF:OBJE>@ /* chain to continued object */	
+1 REFN <USER_REFERENCE_NUMBER>	{0:M}
+2 TYPE <USER_REFERENCE_TYPE>	{0:1}
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}
+1 <<CHANGE_DATE>>	{0:1}

Large whole multimedia objects embedded in a GEDCOM file would break some systems. For this purpose, large multimedia files should be divided into smaller multimedia records by using the subordinate OBJE tag to chain to the next <MULTIMEDIA_RECORD>

fragment. This will allow GEDCOM records to be maintained below the 32K limit for use in systems with limited resources.

Within the definition above, the BLOB subrecord has been marked as deprecated (in GEDCOM 5.5), because on second thoughts, FamilySearch considered BLOBs a bad idea, so they discontinued BLOB support in GEDCOM 5.5.1.
However, it is best to think of this entire GEDCOM 5.5 record as obsolete. Developers should pay attention to the small differences between the GEDCOM 5.5 and 5.5.1 definition, and not simply reuse old code, but make sure that they are using the GEDCOM 5.5.1 definition.

Chain Pointer

The <XREF:OBJE> pointer within the BLOB subrecord (on the line highlighted because it includes an illegal C-style comment) isn't a circular reference. As the GEDCOM 5.5 definition explains, it's there because encoded media file are likely to be larger than the (now obsolete) maximum top-level record size of 32 KB. A GEDCOM 5.5 writer must break the encoded media files into pieces of less than 32KB, and then chain those pieces together. A GEDCOM 5.5 reader must follow the chain pointers and put the encoded media file together again.

GEDCOM 5.5.1 <MULTIMEDIA_LINK>

This is the GEDCOM 5.5.5 <<MULTIMEDIA_LINK>> definition: {1:M}

MULTIMEDIA_LINK:=

n **OBJE** <XREF:OBJE> {1:1}

A <<MULTIMEDIA_LINK>> is a link to a <<MULTIMEDIA_RECORD>> (which is a link to a multimedia file). {0:1}
There should be one multi-media record per file. There may be multiple multi-media links to a multi-media record. {0:1}

That GEDCOM 5.5.5 <<MULTIMEDIA_LINK>> is a significant simplification of the GEDCOM 5.5.1 definition (without page numbers, but with annotations & corrections from the *Annotated Edition*):

MULTIMEDIA_LINK:=

[
n **OBJE** ~~@~~<XREF:OBJE>~~@~~ {1:1}
|
n **OBJE**
~~+1 **FILE** <MULTIMEDIA_FILE_REFN>~~
+1 **FILE** <MULTIMEDIA_FILE_REFERENCE>
+2 **FORM** <MULTIMEDIA_FORMAT>
+3 MEDI <SOURCE_MEDIA_TYPE>
+1 TITL <DESCRIPTIVE_TITLE>
]

Note: some systems may have output the following 5.5 structure. The new context above was introduced in order to allow a grouping of related multimedia files to a particular context.

n OBJE
+1 FILE <MULTIMEDIA_FILE_REFERENCE>
+1 FORM <MULTIMEDIA_FORMAT>
~~+2 MEDI <SOURCE_MEDIA_TYPE>~~
+1 TITLE <DESCRIPTIVE_TITLE>
+1 <<NOTE_STRUCTURE>>

Non-Existent <MULTIMEDIA_FILE_REFN>

The <<MULTIMEDIA_LINK>> syntax references <MULTIMEDIA_FILE_REFN>, but the GEDCOM lineage-linked form doesn't define <MULTIMEDIA_FILE_REFN>, it defines <MULTIMEDIA_FILE_REFERENCE>.

This is the *actual* GEDCOM 5.5 <MULTIMEDIA_LINK> definition:

MULTIMEDIA_LINK:=

[/* embedded form */
n **OBJE** ~~@~~<XREF:OBJE>~~@~~ {1:1}
| /* linked form */

n **OBJE**
+1 FORM <MULTIMEDIA_FORMAT> {1:1}
+1 TITL <DESCRIPTIVE_TITLE> {0:1}
+1 FILE <MULTIMEDIA_FILE_REFERENCE> {1:M}
+1 <<NOTE_STRUCTURE>> {0:M}
]

This structure provides two options in handling the GEDCOM multimedia interface. The first alternative (embedded) includes all of the data, including the multimedia object, within the transmission file. The embedded method includes pointers to GEDCOM records that contain encoded image or sound objects. Each record represents a multimedia object or object fragment. An object fragment is created by breaking the multimedia files into several multimedia object records of 32K or less. These fragments are tied together by chaining from one multimedia object fragment to the next in sequence. This procedure will help manage the size of a multimedia GEDCOM record so that existing systems which are not expecting large multimedia records may discard the records without crashing due to the size of the record. Systems which handle embedded multimedia can reconstitute the multimedia fragments by decoding the object fragments and concatenating them to the assigned multimedia file.

The second method allows the GEDCOM context to be connected to an external multimedia file. This process is only managed by GEDCOM in the sense that the appropriate file name is included in the GEDCOM file in context, but the maintenance and transfer of the multimedia files are external to GEDCOM.

Two <<MULTIMEDIA_LINK>> Forms

The GEDCOM 5.5 and GEDCOM 5.5.1 <<MULTIMEDIA_LINK>> definition allows two alternative forms The first form, called the *embedded form*, uses an <XREF:OBJE>, a link to a <<MULTIMEDIA_RECORD>>, while the second form, called the *linked form*, contains details for an external multimedia file.

The GEDCOM 5.5 <<MULTIMEDIA_LINK>> definition makes it clear that the so-called *embedded form* was meant for embedded multimedia objects only, and GEDCOM 5.5.1 does not support embedded multimedia objects.
The *GEDCOM 5.5.1. Annotated Edition* notes that you might think that this implies that the *embedded form* is obsolete now, but that it is not; on the contrary, it is the preferred form.

The so-called *linked form* of the GEDCOM 5.5.1 <<MULTIMEDIA_LINK>> record offers a subset of the GEDCOM 5.5.1 <<MULTIMEDIA_RECORD>> features. That makes it superfluous, and that is why the *Annotated Edition* deprecated the *linked form*.

The *GEDCOM 5.5.1. Annotated Edition* provides the following best practice for GEDCOM 5.5.1:

It is recommended that all multimedia files be recorded in GEDCOM 5.5.1 using the top-level OBJE record, and that all instances of the <MULTIMEDIA_LINK> record are <XREF:OBJE> links to these <<MULTIMEDIA_RECORD>> records.
Consider the so-called *linked form* of the <<MULTIMEDIA_LINK>> record deprecated.

GEDCOM 5.5.1 Specification misrepresents GEDCOM 5.5 <<MULTIMEDIA_LINK>>

The GEDCOM 5.5.1 <<MULTIMEDIA_LINK>> definition claims that the (linked form of the) GEDCOM 5.5 <<MULTIMEDIA_LINK>> definition (after the indicated corrections) looks like this:

MULTIMEDIA_LINK:=

n OBJE
+1 FILE <MULTIMEDIA_FILE_REFERENCE> {1:M}
+1 FORM <MULTIMEDIA_FORMAT> {1:1}
+2 MEDI <SOURCE_MEDIA_TYPE> {0:1}
+1 TITL <DESCRIPTIVE_TITLE> {0:1}

However, that is not correct. The linked form of the GEDCOM 5.5 <<MULTIMEDIA_LINK>> definition looks like this:

MULTIMEDIA_LINK:=

n OBJE
+1 FORM <MULTIMEDIA_FORMAT> {1:1}
+1 TITL <DESCRIPTIVE_TITLE> {0:1}
+1 FILE <MULTIMEDIA_FILE_REFERENCE> {1:M}
+1 <<NOTE_STRUCTURE>> {0:M}

The GEDCOM 5.5 definition does not include a MEDI subrecord at all, neither as direct subrecord of the OBJE record, nor as a subrecord of the OBJE.FORM record. The GEDCOM 5.5 definition allows notes, while the GEDCOM 5.5.1 definition does not.

Editing Error

The misrepresentation of the GEDCOM 5.5 syntax seems an unintentional editing error. The two differences between the GEDCOM 5.5 and 5.5.1 definitions that the author probably wanted to draw the reader's attention to, are:

- The GEDCOM 5.5 record allows just one file, while the GEDCOM 5.5.1 record allows a group of files
- The GEDCOM 5.5 record has the FORM record as a direct subrecord of the OBJE record, while the GEDCOM 5.5.1 record has the FORM record as a subrecord of the OBJE.FILE record.

Syntax Exception

The real point of the remark is to demand a syntax exception for “some systems” that use GEDCOM 5.5.1, but still create MULTIMEDIA_LINK records in GEDCOM 5.5 format! The implied demand is that GEDCOM 5.5.1 readers should accept GEDCOM 5.5 MULTIMEDIA_LINK records, even though they are not part of GEDCOM 5.5.1.

GEDCOM 5.5.1 specification <<MULTIMEDIA_LINK>> Clairvoyance

The *FamilySearch GEDCOM 5.5.1 specification* states that some systems “may have output” (past tense) the GEDCOM 5.5 structure within GEDCOM 5.5.1 files. That sounds like a helpful annotation, but it is a rather remarkable one.
On the day that FamilySearch introduced GEDCOM 5.5.1, FamilySearch already knew that some applications would use GEDCOM 5.5.1, yet continue to use the GEDCOM 5.5 <MULTIMEDIA_LINK> structure instead of GEDCOM 5.5.1 <MULTIMEDIA_LINK> structure?
Unless the FamilySearch GEDCOM editors own a time machine, something is wrong with their claim, or at least their presentation of it.

PAF GEDCOM

What the GEDCOM 5.5.1 specification really demands, without being open and honest about it, is that third parties are expected to accommodate FamilySearch's unwillingness to follow their own specification in their own product!
FamilySearch's Personal Ancestral File 5.x uses GEDCOM 5.5.1, but still uses the GEDCOM 5.5 <MULTIMEDIA_LINK> structure. So, this FamilySearch statement tells us that, when they were still writing the GEDCOM 5.5.1 specification, FamilySearch had already decided that PAF 5 would use GEDCOM 5.5.1, but keep using the GEDCOM 5.5 <MULTIMEDIA_LINK> structure, and then put this note about “some systems” in to get third parties to support PAF's misbehaviour.

GEDCOM 5.5.1 Writer Best Practice

- Use GEDCOM 5.5.1 records within GEDCOM 5.5.1 files.
- Do not use records from other GEDCOM versions within GEDCOM 5.5.1 (unless they are identical).

GEDCOM 5.5.1 Reader Best Practice

- Because PAF will never be updated again, and still has significant market share, third parties should indeed accept the GEDCOM 5.5 <MULTIMEDIA_LINK> structure within PAF GEDCOM 5.5.1 files - but still issue an error for each occurrence.

GEDCOM 5.5.5 Reader Best Practice

- Demand GEDCOM 5.5.5 records. Reject everything else as a fatal error.

one- and two-digit years illegal

The GEDCOM 5.5.1 specification has this definition of <YEAR>:

> **YEAR:= {Size=3:4}**
>
> A numeric representation of the calendar year in which an event occurred.
> Years 1 through 99 must be padded out to at least 3 digits by using leading zeroes.

Notice that a year is a number that is at least 3 characters long; the FamilySearch GEDCOM specification does not allow years of less than 3 digits. One-digit and two-digit years are illegal.

Three or Four Digits

GEDCOM 3.0 and 4.0 demand that all years be 4 digits long. It was only with GEDCOM 5.0 that FamilySearch started to allow 3-digit dates, even while its *How to record dates* section still instructs the reader to enter a 4-digit date:

> Type the day (one or two characters) first, then the month (capitalize the first three letters of the English name of the month; do not use a period at the end), and then the year (four-character numeric year). The day and month may be omitted if unknown.

The demand that year be at least 3 digits long remains present in all subsequent version of GEDCOM, including GEDCOM 5.6.

No Year less than 100

There is no year zero in the Julian or Gregorian Calendar, the year 1 CE is preceded by the year 1 BCE.
The demand that years must have at least three digits can be interpreted as the FamilySearch GEDCOM specification not allowing the years 1 CE through 99 CE, nor 99 BCE through 1 BCE. This interpretation is certainly encouraged by FamilySearch's *PAF 2.1 Family Records Data Structure Description* (23 Jun 1988), which contains the following definition for an 11-bit year field within a 3-byte date field:

```
YEAR (bits 0-11) - A number between 100 and 2047
```

That's a remarkable definition; it explicitly excludes the years 1 through 99, without giving a reason for doing so.

PAF is based on Ancestral Quest 3.0, and one of the changes listed for Ancestral Quest 3.0 on its version history page is “Allowed for dates to preceed [sic] 100 AD”.

Treat as Date Phrase

The GEDCOM 5.0 specification has this definition of <YEAR>:

> **YEAR:= {Size=3:4, Type=NUMBER}**
>
> A numeric representation of the calendar year in which an event occurred. Years less than 3 digits long will be treated as a number in a phrase.

The second sentence of this definition, “Years less than 3 digits long will be treated as a number in a phrase.”, only occurs in GEDCOM 5.0. It's gone in GEDCOM 5.3, which suggests that

FamilySearch reconsidered, and did not think that particular instruction a good idea.

Still, this definition makes it crystal clear that FamilySearch does not like years of two digits or less, that the minimum length of 3 characters is quite deliberate. FamilySearch does not explain this limitation, nor what to do with years smaller than 100.

This restriction does not exist to combat the tendency of people to abbreviate dates, by leaving off the century. Such a genealogy would be so confusing to read, that the author would quickly start using full years. Moreover, genealogy consistency checks will quickly alert users to such a mistake.
The actual reason is a practical one, revealed by PAF's behaviour on entering short years.

Personal Ancestral File

FamilySearch PAF 5.2.18 from 2002 CE is the closest thing we have to a reference implementation. When you enter a year less than 1000 in PAF, PAF will pad it out to 4 digits by using leading zeroes on screen, and export those leading zeroes to GEDCOM.

Additional PAF behaviour reveals *why* FamilySearch does not like years of less than 3 digits. If you enter a year (without leading zeroes) of 31 or less, PAF does not add leading zeroes, but presents a pop-up message-box instead, with the complaint that “The date you typed is not standard”. That's a rather generic message, one that PAF even pops up for things that GEDCOM allows, but PAF does not support.

PAF only does this when the year is 31 or less, and that reveals the reason why. Not only are GEDCOM date formats quite flexible, users in different locales may also want to format dates differently. PAF simply does not know whether you mean `30 Mar 15` when you type `15 Mar 30`. The demand to use at least three digits for years is to avoid ambiguity between years and days.
Years must be at least three digits long to easily distinguish between days and years.

References

- FamilySearch GEDCOM Specifications
- Ancestral Question Version History: Version 3.0 Features

Alias ALIA

The ALIA record has a remarkable history.
FamilySearch GEDCOM 3.0 and 4.0 defined the ALIA (alias) record as way to record alternate names. FamilySearch GEDCOM 5.0 and later specify that alternate names must be recorded using multiple NAME records. At the same time, FamilySearch GEDCOM 5.0 defined a new ALIA record, namely as a way to record that another INDI record is possibly for the same person.

There was no reason to create confusion by calling this the ALIA record. In fact, there was no reason to create this record type at all, because GEDCOM already supports the <<ASSOCIATION_STRUCTURE>> (ASSO record).

The ASSO record should be used for all relationships that aren't direct family relationships. Possibly duplicate INDI records can be associated with each other using the ASSO record; `possible-duplicate` is a good ASSO.RELA value for that usage.

In practice, many products kept creating ALIA record to record alternate names. All versions of Ancestry.com Family Tree Maker kept creating ALIA records. Software MacKiev Family Tree Maker 2014.1 stopped creating ALIA records; it uses multiple NAME records as it should.

The new ALIA record, a link to another INDI record, has never been used. It only occurs in GEDCOM test files.
GEDCOM 5.5.1 Annotated Edition deprecated the INDI.ALIA record. GEDCOM 5.5.5 removed the INDI.ALIA record. The INDI.ALIA record is illegal now.

GEDCOM 5.5.1 Best Practice

GEDCOM Writer

- Do not use ALIA to record alternate names
- Use multiple NAME records to record alternate names
  - Export the main name first
  - Use additional NAME records to support alternate names
  - Use slashes to delimit the surname on every NAME record
- Do not use INDI.ALIA for linking to other INDI; consider it a deprecated feature
- You can use the ASSO record to link an INDI record to another INDI record

GEDCOM Reader

- Read multiple NAME records containing alternate names
- Assume the first NAME record is the main, preferred name
- Optionally, read INDI.ALIA and INDI.NAME.ALIA records containing alternate names
- If the INDI.ALIA line value appears to be a reference, do not import it, but warn that this legal usage is considered deprecated and not supported

References

- GEDCOM ALIA
- FTM 2017 GEDCOM

NOTE.SOUR.NOTE.SOUR...

The GEDCOM 5.5 errata sheet states that an optional reference to a SOUR record should be added to the NOTE_STRUCTURE definition, like this:

```
[
n NOTE @<XREF:NOTE>@ {1:1}
    +1 SOUR @<XREF:SOUR>@
|
n NOTE [<SUBMITTER_TEXT> | <NULL>] {1:1}
    +1 [CONC|CONT] <SUBMITTER_TEXT> {0:M}
    +1 SOUR @<XREF:SOUR>@
]
```

The GEDCOM 5.5.1 specification does *not* include the additional lines.

Loop

FamilySearch probably realised that this addition was a bad idea, because it allowed notes to have sources, which may have notes, which may sources, and so on, *ad infinitum*, and thus includes the possibility of a loop.

GEDCOM 5.5.1 allows source citations on top-level NOTE records, and allows notes on source records, but does not allow source citation on those SOUR.NOTE note records.

Maximum Path Length

Maximum Length

Arguably, the GEDCOM specification should not impose a maximum length on file paths, but instead merely demand that the file path is legal and - at least in the context of the source system - valid.

The GEDCOM 5.5.1 lineage-linked specification for <MULTIMEDIA_FILE_REFERENCE> states that the file name should be a full path, yet allows only 30 characters. That is not a realistic maximum length for a full path. This is a mistake, plain and simple.

The GEDCOM 5.5.5 changed the maximum to 259 code units, and the GEDCOM 5.5.1 Annotated Edition already strongly suggested that GEDCOM 5.5.1 readers, writers and validators treat the maximum length as 259 characters. The length of 259 characters corresponds to the maximum path length for Windows APIs without long path support; `MAX_PATH -1`, the minus 1 is because `MAX_PATH` includes space for a terminating null character.
This is a legacy limitation; the NTFS file system and Windows APIs with long path support allow paths up to 32K.

CONC and CONT

The maximum <MULTIMEDIA_FILE_REFERENCE> length that will fit on a single CR/LF-terminated GEDCOM line is 247 characters. The GEDCOM grammar supports line values longer than the maximum line length through the CONC tag.
However, because of the misleading explicit inclusion of CONC and CONT tags in the GEDCOM 5.5 and 5.5.1 lineage-linked specification, some GEDCOM 5.5 and 5.5.1 readers still feature limited support for CONC.

A GEDCOM reader that supports CONC correctly, is able to accept file paths of any length. Genealogy software developers whose products do not fully support CONC and CONT yet, are strongly encouraged to fix their GEDCOM readers.

References

MSDN: Naming Files, Paths and Namespaces

The ANSEL Header Demand

This is about a fundamental issue with the FamilySearch GEDCOM 5.5.1 specification, that has been fixed in GEDCOM 5.5.5.
The GEDCOM 5.5.1 <<HEADER>> description contains the following demand (bolding by FamilySearch):

> "All character codes greater than 0x7F **must be converted to ANSEL**"

Catch-22

A GEDCOM writer specifies the character encoding used through the `HEAD.CHAR` line value. A GEDCOM reader must process the GEDCOM file using the specified encoding. There is a catch-22 here: A GEDCOM reader must read the GEDCOM header to discover the character encoding used, but it cannot read the GEDCOM header until it knows the encoding of the file.

FamilySearch "Solution"

FamilySearch's preposterous "solution" to this catch-22 is to demand that the header is encoded using ANSEL, regardless of the encoding the header specifies for the file. That demand is both impossible and impractical.
It is generally impossible because ANSEL supports only a tiny subset of Unicode; most Unicode characters cannot be converted to ANSEL. It is immensely impractical because a GEDCOM reader detects the end of a record by detecting the beginning of the next record, and now it would have to do that while the next record may use a different character encoding - that imposes considerable complexity.
Perhaps the most fundamental issue of all is that if you put an ANSEL-encoded header in front of an UTF-16 encoded GEDCOM file, it is no longer an UTF-16 encoded GEDCOM file. In fact, if you put an ANSEL-encoded header in front of anything but an ANSEL-encoded file, it probably isn't a text file anymore.

By the way, you may be tempted to assume that this ANSEL demand is some holdover from a pre-Unicode version of GEDCOM, but it is not. Unicode support was added in GEDCOM 5.3, and this ANSEL demand was added in GEDCOM 5.4.

Even stranger

FamilySearch surely meant their ANSEL-demand to say that the entire header should be encoded using ANSEL, but what they actually wrote is even stranger; only characters above 0x7F must be converted to ANSEL, so a header for a UTF-16 GEDCOM should use UTF-16 for code points 0x00 through 0x7F, and use ANSEL for all other code points, resulting in an messy mix of UTF-16 and ANSEL encoded characters...

Contradiction

The ANSEL demand made in the <<HEADER>> description is contradicted by GEDCOM 5.5.1 *Chapter 3 Using Character Sets in GEDCOM*:

> If the Unicode environment is used to produce a GEDCOM ~~transmission~~ file, the header record would also be in Unicode, requiring receiving systems to determine whether the ~~transmission~~ GEDCOM file is Unicode or ASCII before they could interpret the GEDCOM header.

That sentence is sloppy in more ways than one, but still clearly communicates that a GEDCOM file encoded in UTF-16 has a GEDCOM header encoded in UTF-16, not ANSEL.

Another sentence in GEDCOM 5.5.1 *Chapter 3 Using Character Sets in GEDCOM* is even clearer:

> The character set for an entire ~~transmission~~ GEDCOM file is specified in the character set line of the header record.

That statement leaves no doubt: the entire file (“transmission” is confused FamilySearch-speak for file) uses one encoding, the one specified in the GEDCOM header.

One Encoding Throughout

GEDCOM files are supposed to be text files. To be text files, they must use a single encoding and single line terminator throughout the file.
The GEDCOM header must use the same encoding as the rest of the file.

Real-World Practice

FamilySearch's defective ANSEL demand is universally ignored. Not one genealogy software developer has ever tried to implement it in any product. All genealogy software uses the encoding specified in the header for the entire file, including the header itself.
Even FamilySearch themselves do not do what FamilySearch demands! FamilySearch's Personal Ancestral File encodes each GEDCOM header the same way as the rest of the GEDCOM file.

Recognising the Encoding

There *is* a catch-22. The key to reading a GEDCOM file correctly is to read the header first, and the key to reading a GEDCOM header correctly is to figure out the encoding first. There is much to say about reading a GEDCOM header correctly, but to keep this brief: the solution to the catch-22 is to start with an analysis pass through the header which only figures out the encoding, before actually reading the header.

GEDCOM Writer Best Practice

- choose one legal encoding
- use that encoding for the entire GEDCOM file
- specify the encoding used through the HEAD.CHAR line value

References

GEDCOM header encoding

HEAD.CHAR.VERS

History

GEDCOM 5.5.1 defined a `HEAD.CHAR.VERS` record, that was, confusedly, supposed to contain a version number for the character encoding specified in `HEAD.CHAR.VERS`. The superfluous `HEAD.CHAR.VERS` record was never used as intended, but has been abused in odd ways, to specify one character set in `HEAD.CHAR` and (not really specify) another character set (instead of a version number) in `HEAD.CHAR.VERS`. The bonus chapter HEAD.CHAR.VERS discusses this in some detail.

GEDCOM 5.5.5 has eliminated this superfluous subrecord from the GEDCOM header; its inclusion in a GEDCOM header is illegal now. This ends the abuse of this superfluous record, and the complexity that abuse imposes on GEDCOM readers trying to the right thing when faced with a wrong header.
GEDCOM writers must specify the character encoding used in `HEAD.CHAR`.

HEAD.CHAR.VERS

The mandatory HEAD.CHAR line value specifies the character encoding used. The optional HEAD.CHAR.VERS is a version number, so presumably specifies a version number for that character encoding.

The HEAD.CHAR.VERS value is never used (but it is abused), and few GEDCOM readers support it.

Support

Character sets can have version numbers, but are designed such that regular applications need not be aware which version they are using. Different versions of Windows support different versions of Unicode, but most Windows developers are not even aware of that.
FamilySearch never made it clear just why they included an optional HEAD.CHAR.VERS record. The most likely reason is simply that they mistakenly assumed that it would be a good idea. The truth is that is completely superfluous.

Real-World HEAD.CHAR.VERS Abusage

The HEAD.CHAR.VERS record has been creatively abused by GEDitCOM, MacFamilyTree and Ahnenblatt.

Reunion

Reunion is a MacOS application that supports the MacRoman character set. Reunion specifies the use of this illegal character set just like any other character set, legal or not, through the HEAD.CHAR line value, like this:

```
1 CHAR MACINTOSH
```

The MacRoman character set should not be used, but when it is used anyway, that is how it should be specified.

GEDitCOM

GEDitCOM is another MacOS application that supports the MacRoman character set.
GEDitCOM specifies the usage of this illegal character set like this:

```
1 CHAR ASCII
2 VERS MacOS Roman
```

This is completely wrong. GEDitCOM starts by lying that the character set used is ASCII. Then, it claims that MacRoman is a particular version of ASCII (it is not), and we are supposed to interpret that claim as the statement that MacRoman is being used.
GEDitCOM should not use MacRoman at all, but when it does, it should be honest about it and specify it in a straightforward, non-confusing way, like Reunion.

This self-contradictory abusage of HEAD.CHAR.VERS does not deserve to be supported.
GEDitCOM itself has to support it, but only because genealogy applications should be able to read GEDCOM and not-quite-GEDCOM files produced by itself, including earlier versions of itself.
If the character set specified is ASCII, the GEDCOM file should be interpreted as ASCII, and processing should end with a fatal error as soon as a non-ASCII code is encountered.

MacFamilyTree

MacFamilyTree up to version 6 or so specifies MacRoman in exactly the same erroneous ways as GEDitCOM does. From around version 6 up to and including version 8.3.4, MacFamilyTree uses a minor variation of the same syntax:

```
1 CHAR ASCII
2 VERS MACINTOSH
```

This was fixed in MacFamilyTree 8.3.5, released in April 2018; MacFamilyTree 8.3.5 and later uses `1 CHAR MACINTOSH` as it should.

Ahnenblatt

This is how Ahnenblatt abuses the HEAD.CHAR.VERS value:

```
1 CHAR ANSI
2 VERS 1252
```

The HEAD.CHAR value specifies that Ahnenblatt is using Windows ANSI. That is an illegal choice, but this is the right way to specify it.
However, the HEAD.CHAR.VERS value below it is nonsense. There is no Windows ANSI version 1252.
The intention is to specify code page 1252, but the HEAD.CHAR.VERS value is not for code pages, it is for version numbers.
What's more, if there were a way to specify the Windows ANSI code page, it would not be necessary to specify code page 1252. There is no way to specify a particular Windows ANSI code page. GEDCOM does not allow Windows ANSI. In practice, Windows ANSI is assumed to be code page 1252.

This HEAD.CHAR.VERS abusage does not deserve to be supported either. Applications should not be using Windows ANSI at all, it is illegal. Readers for GEDCOM 5.5.1 and earlier that do support Windows ANSI, should always assume code page 1252.

Ahnenblatt 3.0 (2019 CE), exports to UTF-8 exclusively.

GEDCOM Reader Best Practice

- read the HEAD.CHAR value
- upon encountering a HEAD.CHAR.VERS record, issue the warning that HEAD.CHAR.VERS is not supported

- report an error if the HEAD.CHAR.VERS line value does not look like a version number (demand at least two numbers separated by a dot)

GEDCOM Writer Best Practice

- use only legal encodings
- specify the encoding used through the HEAD.CHAR value
- do not use the HEAD.CHAR.VERS record to specify a character set version
- do not abuse the HEAD.CHAR.VERS record for anything else either

GEDCOM Validator Best Practice

- upon encountering a HEAD.CHAR.VERS record, warn that it has never been used, only abused
- report an error if the HEAD.CHAR.VERS line value does not look like a version number (demand at least two numbers separated by a dot)

References

- GEDCOM Character Encodings

End of Specification & Bonus

The end of the GEDCOM 5.5.5 specification and Bonus Chapters.