The GEDCOM Standard

7.0.0-rc1

Prepared by the

Family History Department
The Church of Jesus Christ of Latter-day Saints

10 February 2021

Suggestions and Correspondence:

Family History Department
GEDCOM Relations Manager
15 East South Temple Street
Salt Lake City, UT 84150 USA

GEDCOM@ChurchOfJesusChrist.org

Contents

Introduction

GEDCOM was developed by the Family History Department of The Church of Jesus Christ of Latter-day Saints to provide a flexible, uniform format for exchanging computerized genealogical data. GEDCOM is an acronym for **GE**nealogical **D**ata **COM**munication. Its purpose is to foster sharing genealogical information and to develop a wide range of inter-operable 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. This document describes GEDCOM as follows:

- GEDCOM container format (see Chapter 1 (p.7))
- GEDCOM data types (see Chapter 2 (p.19))
- GEDCOM genealogical structures (see Chapter 3 (p.25))

Chapter 1 describes the lower level, known as the **GEDCOM container format**. The GEDCOM container format is a general-purpose data representation language for representing any kind of structured information in a sequential medium, similar to XML, JSON, YAML, or SDLang. Chapter 1 discusses the syntax and identification of structured information in general, but it does not deal with the semantic content of any particular kind of data.

Chapter 2 describes several **data types** used to represent genealogical information, such as a date format that permits dating in multiple calendar systems.

Chapter 3 describes a set of nested **GEDCOM genealogical structures**. This provides structures for representing historical claims, such as individuals, families, and events; sourcing information, such as sources, repositories, and citations; and research metadata, such as information about researchers and rights.

> *Note* — Prior to GEDCOM 7,
>
> - the GEDCOM container format was called the GEDCOM data format.
> - the GEDCOM data types were unnamed and described in various places throughout the document.
> - the GEDCOM genealogical structures were known as the Lineage-Linked GEDCOM Form.

Purpose for Version 7.x

There have been multiple prior releases of GEDCOM, with somewhat idiosyncratic version numbering. The first public comment draft was released in 1984. The previous major version was 5.5.1 which was released in draft status in November 1999 and re-released as a standard in October 2019.

Version 7.0 has a number of goals, including

- Clarify ambiguities in the specification.
- Simplify implementations by removing special-case handling.
- Modernize character encoding, length restrictions, and specification wording.
- Introduce semantic versioning (see https://semver.org/).
- Add better multimedia handling, negative assertions, and rich-text notes.
- Add standard support for common extensions to 5.5.1.
- Provide tools for better interoperability of extensions.

GEDCOM 7.x introduces several breaking changes with GEDCOM 5.5.1; GEDCOM 5.5.1 files are, in general, not valid GEDCOM 7.0 files and *vice versa*. These breaking changes were necessary to remove complicated constructs left over from earlier GEDCOM versions. For a complete list of changes, see the GEDCOM changelog.

A Guide to Version Numbers

Starting with version 7.0.0, GEDCOM version numbers use semantic versioning. The three numbers are titled *major.minor.patch*.

A new *major* version may make arbitrary changes to the specification. Distinct major versions are not in general either forward or backward compatible with one another.

A new *minor* version will preserve the validity of data from all previous minor versions. It may make additional data valid, for example by adding new structure types, allowing current structures in new contexts, or adding new enumerated values or calendars. A minor release will not change the semantic meaning of data from previous minor releases, so for example a 7.0 GEDCOM document is also a valid GEDCOM 7.1 document and represents the same information in both.

A new *patch* version is a clarified or improved specification for the same data and introduces no changes in the GEDCOM data itself. Any software that correctly implements *X.Y.Z* also correctly implements *X.Y.W*. If there is an ambiguity or contradiction in the specification, it will be resolved in a patch version unless it is known that implementations interpreted the spec differently and that clarifying the intended meaning would cause incompatibilities between those implementations.

It is recommended that implementations accept all data at their own or a lesser minor version, regardless of the patch version. It is also recommended that they import data from subsequent minor versions by treating any unexpected structures, enumerations, or calendars as if they were extensions (p.13).

1. GEDCOM container format

1.1. Characters

Each GEDCOM document is a sequence of octets or bytes. The octets encode a sequence of characters according to the UTF-8 character encoding as described in §9.2 of ISO/IEC 10646.

The first character in each GEDCOM document *should* be U+FEFF, the byte-order mark. If present, this initial character has no meaning within GEDCOM but serves to indicate to other systems that GEDCOM uses the UTF-8 character encoding.

Certain characters *must not* appear in any GEDCOM data:

- The C0 control characters other than tab and line endings (U+0000–U+001F except U+0009, U+000A and U+000D)
- The DEL character (U+007F)
- Surrogates (U+D800–U+DFFF)
- Invalid code points (U+FFFE and U+FFFF)

Implementations should be aware that bytes per character and characters per glyph are both variable when using UTF-8. Use of Unicode-aware processing and display libraries is *recommended*.

Character-level GEDCOM grammar is specified in this document using Augmented Bakaus-Naur Form (ABNF) as defined in IETF STD 68 (https://tools.ietf.org/html/std68) and modified in IETF RFC 7405 (https://tools.ietf.org/html/rfc7405). The banned characters can be expressed in ABNF as production banned (p.7):

```
banned = %x00-08 / %x0B-0C / %x0E-1F ; C0 other than LF CR and Tab
       / %x7F                        ; DEL
       / %x80-9F                     ; C1
       / %xD800-DFFF                 ; Surrogates
       / %xFFFE-FFFF                 ; invalid
```

All other ABNF expressions in this document assume the absence of any characters matching production banned.

This document additionally makes use of the following named character sets in ABNF:

```
digit       = %x30-39   ; 0 through 9
nonzero     = %x31-39   ; 1 through 9
ucletter    = %x41-5A   ; A through Z
underscore  = %x5F      ; _
atsign      = %x40      ; @
```

1.2. Structures

A GEDCOM **structure** consists of a structure type, an optional **payload**, and a collection of substructures. The payload belongs to one of several data types, as described in Chapter 2 (p.19).

Every structure is either a **record**, meaning it is not contained in any other structure's collection of substructures, or it is a **substructure** of exactly one other structure. The other structure is called its **superstructure**. Each substructure either refines the meaning of its superstructure, provides metadata about its superstructure, or introduces new data that is closely related to its superstructure.

The collection of substructures is partially ordered. Substructures with the same structure type are in a fixed order, but substructures with different structure types may be reordered. The order of substructures of a single type indicates user preference, with the first substructure being the most-preferred value, unless a different meaning is explicitly indicated in the structure's definition.

A structure must have either a non-empty payload or at least one substructure. Empty payloads and missing payloads are considered equivalent. The remainder of this document uses "payload" as shorthand for "non-empty payload".

> *Note* — Unlike structures, pseudo-structures needn't have either payloads or substructures. `TRLR` (p.76) never has either, and `CONT` (p.60) doesn't when payloads contain empty lines.

A structure is a representation of data about its **subject**. Examples include the entity, event, claim, or activity which the structure describes.

GEDCOM documents also contain three types of pseudo-structures:

- The header resembles a record, comes first in each document, and contains metadata about the entire document in its substructures. See The Header (p.13) for more.
- The trailer resembles a record, comes last in each document, and cannot contain substructures.
- A line continuation resembles a substructure, comes before any other substructures, is used to encode multi-line payloads, and cannot contain substructures.

Previous versions of GEDCOM limited the number of characters that could appear in a structure, record, and payload. Those restrictions were removed in GEDCOM 7.0.

1.3. Lines

A GEDCOM **line** is a string representation of (part of) a GEDCOM *structure*. A line consists of a level, optional cross-reference identifier, tag, optional line value, and line terminator. It matches the production `Line`:

```
Line    = Level D [Xref D] Tag [D LineVal] EOL

EOL     = %x0D [%x0A] / %x0A              ; CR-LF, CR, or LF
D       = %x20                            ; space
Tag     = stdTag / extTag
Xref    = atsign 1*tagchar atsign         ; but not "@VOID@"
LineVal = pointer / lineStr
Level   = "0" / nonzero *digit

tagchar = ucletter / digit / underscore
extTag  = underscore 1*tagchar
stdTag  = ucletter *tagchar

Pointer = voidPtr / Xref
voidPtr = %s"@VOID@"

nonAt   = %x09 / %x20-3F / %x41-10FFFF    ; non-EOL, non-@
nonEOL  = %x09 / %x20-10FFFF              ; non-EOL
lineStr = (nonAt / atsign atsign) *nonEOL ; leading @ doubled
```

The **level** matches production `Level` (p.9) and is used to encode substructure relationships. Any line with level 0 encodes a record or a record-like pseudo-structure. Any line with level $x > 0$ encodes a substructure of the structure encoded by the nearest preceding line with level $x - 1$.

The **cross-reference identifier** matches production `Xref` (but not `voidPtr`) and indicates that this is a structure to which pointer-type payloads may point. Each cross-reference identifier must be unique within a given data document. Cross-reference identifiers are not retained between transmissions and *should not* be made visible to the user to avoid them referring to transient data within notes or other durable data.

Each record to which other structures point *must* have a cross-reference identifier. A record to which no structures point *may* have a cross-reference identifier, but does not need to have one. A substructure or pseudo-structure *must not* have a cross-reference identifier.

The **tag** matches production `Tag` and encodes the structure's type. Tags that match the production `stdTag` are defined in this document. Tags that match `extTag` are defined according to Extensions (p.13).

The **line value** matches production `LineVal` and encodes the structure's payload. Line value content is sufficient to distinguish between pointers and non-pointers. Pointers are encoded as the cross-reference identifier of the pointed-to structure. The exact encoding of non-pointer payloads is dependent on the datatype of the payload, as determined by the structure type. The datatype of non-pointer payloads cannot be fully determined by line value content alone.

If a line value matches production `Xref`, the same value *must* occur as the cross-reference identifier of a structure within the document. The special `voidPtr` production is provided to encode null pointers.

A leading `@` (U+0040) in a non-pointer payload *shall* be escaped in the corresponding line value by doubling the `@`.

> *Note* — Line values that match neither `Xref` nor `lineStr` are prohibited in this version of GEDCOM. They have been used in previous versions of GEDCOM (for example, a line value beginning `@#D` was a date in GEDCOM 4.0 through 5.5.1) and may be used again in a future version of GEDCOM if an appropriate need arises.

The components of a line are each separated by a single **delimiter** matching production `D`. A delimiter is always a single space character (U+0020). Using multiple delimiters between components of a line is prohibited. Thus if the tag is followed by two spaces, the first space is a delimiter and the second space is part of the line value.

All characters in a payload must be preserved in the corresponding line value, including preserving any leading or trailing spaces.

Each line is ended by a **line terminator** matching production `EOL`. A line terminator may be a carriage return U+000D, line feed U+000A, or a carriage return followed by a line feed. The same line terminator *should* be used on every line of a given document.

Line values cannot contain internal line terminators, but some payloads can. If a payload contains a line terminator, the payload is split on the line terminators into several payloads. The first of these split payloads is encoded as the line value of the structure's line, and each subsequent split payload is encoded as the line value of a **line continuation** pseudo-structure of the structure. The tag of a line continuation pseudo-structure is `CONT` (p.60). The order of the line continuation pseudo-structures matches the order of the lines of text in the payload.

Line continuation pseudo-structures are not considered to be structures nor to be part of a structure's collection of substructures. They *must* appear immediately following the line whose payload they are encoding and before any other line.

Because line terminators in payloads are encoded using line continuations, it is not possible to distinguish between U+000D and U+000A in payloads.

Previous versions of GEDCOM limited the number of characters that could appear in a tag, cross-reference identifier, and line-value. Those restrictions were removed in GEDCOM 7.0. The `CONC` pseudo-structure, which allowed line values to have a shorter length restriction than payloads, was also removed.

Previous versions of GEDCOM allowed spaces, tabs, and line terminators to precede the level of a line. That permission was removed from GEDCOM 7.0 to simplify parsing.

Previous versions of GEDCOM required doubling all `@` in a line value, but such doubling was not widely implemented in practice. Only an initial `@` is doubled in GEDCOM 7.0.

Example — The following are examples of valid but unrelated GEDCOM lines:

- level 0, cross-reference identifier `@I1234@`, tag `INDI` (p.66), no line value.

```
0 @I1234@ INDI
```

- level 1, no cross-reference identifier, tag `CHIL` (p.59), pointer line value pointing to the structure with cross-reference identifier "`@I1234@`".

```
1 CHIL @I1234@
```

- level 1, no cross-reference identifier, tag `NOTE` (p.69), and line value + continuation pseudo-structure to encode a four-line payload string: "`This is a note field that`", "`spans four lines.`", "", and "`(the third line was blank)`".

```
1 NOTE This is a note field that
2 CONT spans four lines.
2 CONT
2 CONT (the third line was blank)
```

1.4. The Header and Trailer

Every GEDCOM document *must* begin with a Header pseudo-structure and end with a trailer pseudo-structure.

The trailer pseudo-structure has level `0`, tag `TRLR` (p.76) and no line value or substructures. The trailer has no semantic meaning; it is present only to mark the end of the GEDCOM document.

The header pseudo-structure has level `0`, tag `HEAD` (p.66), and no line value. The substructures of the header pseudo-structure provide metadata about the entire GEDCOM document. Some of those substructures are defined here; others are defined in Chapter 3 (p.25) or by extensions.

Every header must contain a substructure with a known tag that identifies the standard to which the GEDCOM document complies. For the GEDCOM 7.0 genealogical structures, this is the `GEDC` (p.65) structure described in Chapter 3 (p.65).

A header *should* contain an extension schema structure with tag `SCHMA` (p.73) as described in Extensions (p.13).

1.5. Extensions

A **standard structure** is a structure whose type, tag, meaning, superstructure, and cardinality within the superstructure are described in this document. This includes records such as `INDI` (p.66), documented substructures such as `INDI`.`NAME`, and payload-type substructures such as `INDI`.`CHAN`.`TIME`.

Two forms of **extension structures** are permitted:

- A **tagged extension structure** is a structure whose tag matches production `extTag` (p.9). Tagged extension structures may appear as records or substructures of any other structure.
- An **extended-use standard structure** is a structure whose type, tag, and meaning are defined in this document and whose superstructure is a tagged extension structure.

Extension structures may have substructures, which may be either tagged extension structures of extended-use standard structures.

All other non-standard structures are prohibited. Examples of prohibited structures include, but are not limited to,

- any structure with a tag matching production `stdTag` that is not defined in this document;
- any substructure with cardinality `{0:1}` appearing more than once;
- a standard substructure appearing as a record or vice-versa;
- a standard structure whose payload does not match the requirements of this document.

> *Note* — In some cases, an extension may wish to allow multiple structures where this document allows only one. The recommended way to do this is to create an extension tag and URI and serve a page describing how the semantics of the structure have been extended to allow multiple instances.
>
> > *Example* — Suppose I have multiple records that give different ages of the wife at a wedding; however, this standard allows only one `MARR` (p.53).`WIFE`.`AGE`. An extension could not include multiple `MARR`.`WIFE` nor `MARR`.`WIFE`.`AGE`, but could define a new extension `_AGE`, give it a URL, and provide the following definition of this extension structure type at that URL:
> >
> > > Alternate age: an age attested by some source, but not accepted by the researcher as the actual age of the individual. If the age is accepted by the researcher, the standard tag `AGE` (p. 58) should be used instead
> >
> > then provide the following data
> >
> > ```
> > 1 MARR
> > 2 WIFE
> > 3 AGE 27y
> > 3 _AGE 22y
> > ```

Enumerated values may be extended with new values that match production `extTag` (p.9). Enumerations may not use standard values from other enumeration sets.

Example — The following is not allowed because `PARENT` is defined as a value for `ROLE` (p.72), not for `RESN` (p.72)

```
0 @BAD@ INDI
1 RESN PARENT
1 NOTE The above enumeration value is not allowed
```

Dates may be extended provided they use a calendar that matches production `extTag`. Dates with extension calendars may also use extension months and epochs.

1.5.1. Extension Tags

Each use of the `extTag` production is called an extension tag, including when used as a tag, calendar, month, epoch, or enumerated value. Each `extTag` is either a *documented extension tag* or an *undocumented extension tag*. It is recommended that documented extension tags be used instead of undocumented extension tags wherever possilbe.

A **documented extension tag** is one that is mapped to a URI using the schema structure. The schema structure is a substructure of the header with tag `SCHMA` (p.73). It should appear within the document before any extension tags. The schema's substructures are tag definitions.

A tag definition is a structure with tag `TAG` (p.74). Its payload is an extension tag, a space, and a URI and defines that extension tag to be an abbreviation for that URI within the current document.

Example — The following GEDCOM header

```
0 HEAD
1 SCHMA
2 TAG _SKYPEID http://xmlns.com/foaf/0.1/skypeID
2 TAG _MEMBER http://xmlns.com/foaf/0.1/member
```

defines the following tags

Tag	Means
`_SKYPEID`	`http://xmlns.com/foaf/0.1/skypeID`
`_MEMBER`	`http://xmlns.com/foaf/0.1/member`

The meaning of a documented extension tag is identified by its URI, not its tag. Documented extension tags can be changed freely by modifying the schema, though it is recommended that documented extension tags not be changed. However, a tag change may be necessary if a product picks the same tags for URIs that another product uses for different URIs.

> *Example* — The following two GEDCOM documents are semantically equivalent and a system importing one may export it as the other without change of meaning.
>
> ```
> 0 HEAD
> 1 SCHMA
> 2 TAG _SKYPEID http://xmlns.com/foaf/0.1/skypeID
> 0 @I0@ INDI
> 1 _SKYPEID example.person
> ```
>
> ```
> 0 HEAD
> 1 SCHMA
> 2 TAG _SI http://xmlns.com/foaf/0.1/skypeID
> 0 @I0@ INDI
> 1 _SI example.person
> ```

An extension tag that is not given a URI in the schema structure is called an **undocumented extension tag**. The meaning of an undocumented extension tag is identified by its tag.

1.5.2. Requirements and Recommendations

- It is *recommended* that applications not use undocumented extension tags.
- It is *required* that each tag definition's extension tag be unique within the document.
- It is *recommended* that each documented extension tag's URI be unique within the document.
- It is *recommended* that extension creators use URLs as their URIs and serve a page describing the meaning of an extension at its URL.
- It is *recommended* that extensions use extended-use standard structures instead of tagged extension structures if extended-use standard structures will suffice.

Future versions of GEDCOM may include additional recommendations relating to documentation, machine-readable documentation, or embedded metadata about extensions within the schema.

1.5.3. Extension vs Standard

In all cases, standard structures take priority over extensions. In particular, those supporting extensions should keep in mind the following:

- If a standard structure is present that contradicts an extension that is present, the standard structure has priority and the extension should be updated to align with it.

  > *Example* — If a document has an extension `_ISODATE` in ISO 8601 format that disagrees with a `DATE` (p.61) in the `DateValue` (p.20) format, the `DATE` shall be taken as more correct and the `_ISODATE` updated to reflect that.

- If a standard structure can be extracted as a subset of the semantics of an extension, the standard tag must be generated along with the extension and kept in sync with it by systems understanding the extension.

  > *Example* — If a document has an extension `_LOC` providing a detailed hierarchical place representation with historical names, boundaries, and the like, it must also generate the corresponding `PLAC` (p.71) structures with the subset of that information which `PLAC` can represent.

- If an extension can be extracted as a subset of the semantics of a standard structure, or if the extension and standard structure only sometimes align, then the standard structure should be included if and only if the semantics align in this case.

  > *Example* — If a document has an extension `_PARTNER` that generalizes `HUSB` (p.66) and `WIFE` (p.78) and some `ASSO` (p.59) `ROLE` (p.72)s, then it should pair the extension with those standard structures if and only if it knows which one applies.

  > *Example* — If a document has an extension `_HOUSEHOLD` that is the same as `FAM` (p.63) in some situations but not in others, then it should keep the `_HOUSEHOLD` and `FAM` in sync if and only if they align.

1.6. Removing data

There may be situations where data needs to be removed from a GEDCOM document, such as when a user requests its deletion or marks it as confidential and not for export.

In general, removed data should result in removed structures.

Pointers to a removed structure should be replaced with `voidPtr` (p.9)s.

If removal of a structure makes the superstructure invalid because the superstructure required the substructure, the structure should instead be retained and have its payload changed to a `voidPtr` if a pointer, or to a datatype-appropriate empty value if a non-pointer.

If removing a structure leaves its superstructure with no payload and no substructures, the superstructure should also be removed.

2. GEDCOM data types

Every line value (with any continuation pseudo-structures) is a string. However, those strings can encode one of several conceptual datatypes.

2.1. Text

A free-text string is text in a human language. Conceptually, it may be either a user-generated string or a source-generated string. Programmatically, both are treated as unconstrained sequences of characters with an associated language.

2.2. Integer

An integer is a non-empty sequence of ASCII decimal digits and represents a non-negative integer in base-10. Leading zeros have no semantic meaning and *should* be omitted.

```
Integer = 1*digit
```

Negative integers are not supported by GEDCOM.

2.3. Enumeration

An enumeration is a selection from a set of options. They are represented in GEDCOM as a string matching the same production as a tag, including the rules about extensions beginning with `_` (U+005F) and being mapped to URIs by a schema.

```
Enum     = Tag
```

Each enumeration value has a distinct meaning as identified by its corresponding URI.

2.4. Date

GEDCOM dates are a somewhat involved datatype, including the ability to store approximate dates, date periods, and dates expressed in different calendars.

Technically, there are three distinct date datatypes: `DateValue` is a generic type that can express many kinds of dates. `DateExact` is used for timestamps and other fully-known dates. `DatePeriod` is used to express time intervals that span multiple days.

```
DateValue   = date / DatePeriod / dateRange / dateApprox
DateExact   = day D month D year  ; in Gregorian calendar
DatePeriod  = %s"FROM" D date [D %s"TO" D date]
            / %s"TO" D date

date        = [calendar D] [[day D] month D] year [D epoch]
dateRange   = %s"BET" D date D %s"AND" D date
            / %s"AFT" D date
            / %s"BEF" D date
dateApprox  = (%s"ABT" / %s"CAL" / %s"EST") D date

dateRestrict = %s"FROM" / %s"TO" / %s"BET" / %s"AND" / %s"BEF"
            / %s"AFT" / %s"ABT" / %s"CAL" / %s"EST" / %s"BCE"

calendar = %s"GREGORIAN" / %s"JULIAN" / %s"FRENCH_R" / %s"HEBREW"
         / extTag

day     = Integer
year    = Integer
month   = stdTag / extTag  ; constrained by calendar
epoch   = %s"BCE" / extTag ; constrained by calendar
```

In addition to the constraints above:

- The allowable `month` (p.20)s and `epoch`s are determined by the `calendar`.
- No calendar names, months, or epochs match `dateRestrict`.
- Extension calendars (those with `extTag` (p.9) for their `calendar`) must use `extTag`, not `stdTag`, for months.

An absent `calendar` is equivalent to the calendar `GREGORIAN` (p.88).

The grammar above allows for `date`s to be preceded by various words. The meaning of these words is given as follows:

Production	Meaning
`FROM` *x*	Lasted for multiple days, beginning on *x*
`TO` *x*	Lasted for multiple days, ending on *x*
`BET` *x* `AFT` *x*	Exact date unknown, but no earlier than *x*
`AND` *x* `BEF` *x*	Exact date unknown, but no later than *x*
`ABT` *x*	Exact date unknown, but near *x*
`CAL` *x*	*x* is calculated from other data
`EST` *x*	Exact date unknown, but near *x*; and *x* is calculated from other data

Known calendars and tips for handling dual dating and extension calendars are given in Appendix A: Calendars and Dates (p.88)

Date payloads may also be omitted entirely if no suitable form is known but a substructure (such as a `PHRASE` (p.70) or `TIME` (p.75)) is desired.

> *Note* — GEDCOM 5.3 through 5.5.1 allowed phrases inside `DateValue` payloads. Date phrases were moved to the `PHRASE` substructure in 7.0.

> *Note* — As defined by the grammar above, every date *must* have a year. If no year is known, the entire date may be omitted
>
> > *Example* — The following is an appropriate way to handle a missing year
> >
> > ```
> > 2 DATE
> > 3 PHRASE 5 January (year unknown)
> > ```

2.5. Time

Time is represented on a 24-hour clock (for example, 23:00 rather than 11:00 PM). It may be represented either in event-local time or in Coordinated Universal Time (UTC). UTC is indicated by including a `Z` (U+005A) after the time value; event-local time is indicated by its absence.

```
Time     =  hour ":" minute [":" second ["." fraction]] [%s"Z"]

hour     = digit / ("0" / "1") digit / "2" ("0" / "1" / "2" / "3")
minute   = ("0" / "1" / "2" / "3" / "4" / "5") digit
second   = ("0" / "1" / "2" / "3" / "4" / "5") digit
fraction = 1*digit
```

> *Note* — The above grammar prohibits end-of-day instant `24:00:00` and leap-seconds. It allows both `02:50` and `2:50` as the same time.

2.6. Age

Ages are represented by counts of years, months, weeks, and days.

```
Age         = [ageBound D] ageDuration

ageBound    = "<" / ">"
ageDuration = years [D months] [D weeks] [D days]
            / months [D weeks] [D days]
            / weeks [D days]
            / days

years    = Integer %x79     ; 35y
months   = Integer %x6D     ; 11m
weeks    = Integer %x77     ; 8w
days     = Integer %x64     ; 21d
```

Where

Production	Meaning
`<`	The real age was less than the provided age
`>`	The real age was greater than the provided age
`years`	a number of years
`months`	a number of months
`weeks`	a number of weeks
`days`	a number of days

Non-integer numbers should be rounded down to an integer. Thus, if someone has lived for 363.5 days, their age might be written as `363d`, `51w 6d`, `51w`, `0y`, etc.

Because numbers are rounded down, `>` effectively includes its endpoint; that is, the age `> 8d` includes people who have lived 8 days + a few seconds.

Different cultures count ages differently. Some increment years on the anniversary of birth and others at particular seasons. Some round to the nearest year, others round years down, others up. Because users may be unaware of these traditions or may fail to convert them to the round-down convention used in GEDCOM, errors in age of up to a year are common.

Age payloads may also be omitted entirely if no suitable form is known but a substructure (such as a `PHRASE` (p.70)) is desired.

> *Note* — GEDCOM 5.5 and 5.5.1 allowed a few specific phrases inside `Age` (p.22) payloads. Age phrases were moved to the `PHRASE` substructure in 7.0.

2.7. List

A list is a meta-syntax representing a sequence of values with another datatype. Lists are serialized in a comma-separated form, delimited by one comma (U+002C `,`) and any number of spaces (U+0020) between each item. It is recommended that a comma-space pair (U+002C U+0020) be used as the delimiter.

If valid for the underlying type, empty strings may be included in a list by having no characters between delimiters.

> *Example* — A `List:Text` with value "`, , one, more,`" has five `Text`-type values: two empty strings, the string "`one`", the string "`more`", and one more empty string.

There is no escaping mechanism to allow lists of entries that begin or end with spaces or that contain comma characters.

2.8. Personal Name

A personal name is mostly free-text. It *should* be the name as written in the culture of the individual and *should not* contain line breaks, repeated spaces, or characters not part of the written form of a name (except for U+002F as explained below).

```
NamePersonal = nameStr
             / [nameStr] "/" [nameStr] "/" [nameStr]

nameChar     = %x20-2E / %x30-10FFFF  ; any but '/' and '\t'
nameStr      = 1*nameChar
```

The character U+002F (`/`, slash or solidus) has special meaning in a personal name, being used to delimit the portion of the name that most closely matches the concept of a surname, family name, or the like. This version of GEDCOM does not provide any standard way of representing names that contain U+002F.

2.9. Special

The special datatype is a string conforming to a some case-specific standard or constraints. The constraints on each special datatype instance are either unique to that structure type or are not simply expressed. For example, the payload of a `IDNO` (p.54) structure may obey different rules for each possible `TYPE` (p.76) substructure.

3. GEDCOM genealogical structures

3.1. Introduction

This chapter describes a set of structure types for exchanging family-based lineage-linked genealogical information in the GEDCOM format. Lineage-linked data pertains to individuals linked in family relationships across multiple generations.

The GEDCOM genealogical structures defined in this chapter are based on the general framework of the GEDCOM container format and GEDCOM data types defined in Chapters 1 and 2.

Historically, the GEDCOM genealogical structures were used as the only form approved for exchanging data with Ancestral File, TempleReady and other Family History resource files. Those systems were all replaced between 1999 and 2019, and GEDCOM-X (https://gedcomx.org) was introduced as the new syntax for communication with their replacements. GEDCOM and GEDCOM-X have similar expressive power, but as of 2020 GEDCOM is more common for exchanging single-researcher files between applications and GEDCOM-X is more common for transferring bulk data and communication directly between applications.

3.1.1. Organization

The basic description of the GEDCOM genealogical structures' organization is presented in the following 3 major sections:

- "Structure Organization (p.27)" describes records and other nested structures.
- "Structure Meaning (p.51)" provides a definition of each structure by its tag.
- "Enumeration Values (p.79)" provides a definition of each enumeration value by its containing structure.

3.1.2. A Metasyntax for Structure Organization

The structures, with their payloads and substructures, are represented using a custom metasyntax. The intent of this metasyntax is to resemble the line encoding of allowable structures. In the metasyntax:

- Options are placed between brackets `[` and `]` and have choices separated by pipes `|`.
- Named sets of rules are indicated with a name followed by `:=`.
- Level markers are used to indicate substructure relationships.
  - `0` means "must be a record".
  - `n` means "level inherited from rule instantiation".
  - `+1`, `+2`, etc, indicate nesting within nearest preceding structure with lesser level.
- Rule instantiation is indicated by double angle-brackets (i.e. `<<`rule name`>>`).
- Line templates have several parts:
  - An optional cross-reference template `@XREF:`tag`@`, meaning this structure may be pointed to by other structures.

    Structures that are not pointed to by other structures need not have a cross-reference identifier (p.9) even if their line template has a cross-reference template.
  - The standard tag for this structure.
  - A payload, which is one of the following:
    - `@<XREF:`tag`>@` means a pointer to a structure with this cross-reference template.
    - `<`datatype`>` means a non-pointer payload, as described in GEDCOM data types (p.19)

      Note that some datatypes (`Age` (p.22), `DateValue` (p.20), `DateExact`, `DateRange`, and `Enum` (p.19)) also define permitted substructures for all structures with that payload type. Those substructures are not replicated in the metasyntax.
    - Nothing, meaning this structure does not have a payload.
    - `[`text`|<NULL>]` means the payload is optional but if present must be the given text

- Four cardinality markers are used:
  - `{0:1}` means "optional" – zero or one allowed, and a second must not be present.
  - `{1:1}` means "required" – one *must* be present, and a second must not be present.
  - `{0:M}` means "any number allowed". Unless otherwise specified, the first is the most-preferred value. For example, the display name would be the first name structure, the profile image the first image structure.
  - `{1:M}` means "at least one required". Unless otherwise specified, the first is the most-preferred value.

  Systems interested in violating the cardinality rules should instead create extension structures (p.13) with different cardinality.

Due to the history of GEDCOM's design, the context of a structure's superstructure may be necessary in addition to the structure's standard tag to fully determine its structure type. To refer to a structure in the context of its superstructure, tags are written with intervening periods. For example, `GEDC`.`VERS` refers to a structure with tag `VERS` (p.78) and a superstructure with tag `GEDC` (p. 65).

3.2. Structure Organization

3.2.1. Document

GEDCOM Document :=

```
0 <<HEADER>>                                  {1:1}
0 <<RECORD>>                                  {0:M}
0 TRLR                                        {1:1}
```

`RECORD` :=

```
[
n <<FAMILY_RECORD>>                                {1:1}
|
n <<INDIVIDUAL_RECORD>>                            {1:1}
|
n <<MULTIMEDIA_RECORD>>                            {1:1}
|
n <<REPOSITORY_RECORD>>                            {1:1}
|
n <<SHARED_NOTE_RECORD>>                           {1:1}
|
n <<SOURCE_RECORD>>                                {1:1}
|
n <<SUBMITTER_RECORD>>                             {1:1}
]
```

`HEADER` :=

```
n HEAD                                                  {1:1}
  +1 GEDC                                               {1:1}
     +2 VERS <Special>                                  {1:1}
  +1 SCHMA                                              {0:1}
     +2 TAG <Special>                                   {0:M}
  +1 SOUR <Special>                                     {0:1}
     +2 VERS <Special>                                  {0:1}
     +2 NAME <Text>                                     {0:1}
     +2 CORP <Text>                                     {0:1}
        +3 <<ADDRESS_STRUCTURE>>                        {0:1}
        +3 PHON <Special>                               {0:M}
        +3 EMAIL <Special>                              {0:M}
        +3 FAX <Special>                                {0:M}
        +3 WWW <Special>                                {0:M}
     +2 DATA <Text>                                     {0:1}
        +3 DATE <DateExact>                             {0:1}
           +4 TIME <Time>                               {0:1}
        +3 COPR <Text>                                  {0:1}
  +1 DEST <Special>                                     {0:1}
  +1 DATE <DateExact>                                   {0:1}
     +2 TIME <Time>                                     {0:1}
  +1 SUBM @<XREF:SUBM>@                                 {0:1}
  +1 COPR <Text>                                        {0:1}
  +1 LANG <Special>                                     {0:1}
  +1 PLAC                                               {0:1}
     +2 FORM <List:Text>                                {1:1}
  +1 <<COMMENT_STRUCTURE>>                              {0:1}
```

The header pseudo-structure provides metadata about the entire GEDCOM document. A few substructures of note:

- `GEDC` (p.65) gives the GEDCOM standard this document conforms to.
- `SCHMA` (p.73) gives the meaning of extension tags; see Extensions (p.13) for more.
- `SOUR` (p.74) describes the originating software.
  - `SOUR`.`CORP` describes the corporation creating the software.
  - `SOUR`.`DATA` describes a larger database this data is extracted from.
- `LANG` (p.66) and `PLAC` (p.71) give a default value for the rest of the document.

3.2.2. Records

FAMILY_RECORD :=

```
n @XREF:FAM@ FAM                                   {1:1}
  +1 RESN <List:Enum>                              {0:1}
  +1 <<FAMILY_ATTRIBUTE_STRUCTURE>>                {0:M}
  +1 <<FAMILY_EVENT_STRUCTURE>>                    {0:M}
  +1 <<NON_EVENT_STRUCTURE>>                       {0:M}
  +1 HUSB @<XREF:INDI>@                            {0:1}
     +2 PHRASE <Text>                              {0:1}
  +1 WIFE @<XREF:INDI>@                            {0:1}
     +2 PHRASE <Text>                              {0:1}
  +1 CHIL @<XREF:INDI>@                            {0:M}
     +2 PHRASE <Text>                              {0:1}
  +1 <<ASSOCIATION_STRUCTURE>>                     {0:M}
  +1 SUBM @<XREF:SUBM>@                            {0:M}
  +1 <<LDS_SPOUSE_SEALING>>                        {0:M}
  +1 <<IDENTIFIER_STRUCTURE>>                      {0:M}
  +1 <<CHANGE_DATE>>                               {0:1}
  +1 <<CREATION_DATE>>                             {0:1}
  +1 <<NOTE_STRUCTURE>>                            {0:M}
  +1 <<SOURCE_CITATION>>                           {0:M}
  +1 <<MULTIMEDIA_LINK>>                           {0:M}
```

The `FAM` (p.63) record was originally structured to represent families where a male `HUSB` (p.66) (husband or father) and female `WIFE` (p.78) (wife or mother) produce `CHIL` (p.59) (children). The `FAM` record may also be used for cultural parallels to this, including nuclear families, marriage, cohabitation, fostering, adoption, and so on, regardless of the gender of the partners. Sex, gender, titles, and roles of partners should not be inferred based on partner the `HUSB` or `WIFE` structure points to.

The individuals pointed to by the `HUSB` and `WIFE` are collectively referred to as "partners", "parents" or "spouses".

Some displays may be unable to display more than two partners. Displays may use `HUSB` and `WIFE` as layout hints, for example, by consistently displaying the `HUSB` on the same side of the `WIFE` in a tree view. It is recommended that family structures with more than two partners either use several `FAM` records or use `<<ASSOCIATION_STRUCTURE>>`s to indicate additional partners.

> *Note* — The `FAM` record will be revised in a future version of GEDCOM to fully express the diversity of human family relationships.

The preferred order of the `CHIL` (children) pointers within a `FAM` (family) structure is chronological by birth. A `CHIL` with a `voidPtr` (p.9) indicates a placeholder for an unknown child in this birth order.

If a `FAM` record uses `HUSB` or `WIFE` to point to an `INDI` (p.66) record, the `INDI` record *must* use `FAMS` (p.64) to point to the `FAM` record. If a `FAM` record uses `CHIL` to point to an `INDI` record, the `INDI` record *must* use a `FAMC` (p.64) to point to the `FAM` record.

INDIVIDUAL_RECORD :=

```
n @XREF:INDI@ INDI                                {1:1}
  +1 RESN <List:Enum>                             {0:1}
  +1 <<PERSONAL_NAME_STRUCTURE>>                  {0:M}
  +1 SEX <Enum>                                   {0:1}
  +1 <<INDIVIDUAL_ATTRIBUTE_STRUCTURE>>           {0:M}
  +1 <<INDIVIDUAL_EVENT_STRUCTURE>>               {0:M}
  +1 <<NON_EVENT_STRUCTURE>>                      {0:M}
  +1 <<LDS_INDIVIDUAL_ORDINANCE>>                 {0:M}
  +1 FAMC @<XREF:FAM>@                            {0:M}
     +2 PEDI <Enum>                               {0:1}
        +3 PHRASE <Text>                          {0:1}
     +2 STAT <Enum>                               {0:1}
        +3 PHRASE <Text>                          {0:1}
     +2 <<NOTE_STRUCTURE>>                        {0:M}
  +1 FAMS @<XREF:FAM>@                            {0:M}
     +2 <<NOTE_STRUCTURE>>                        {0:M}
  +1 SUBM @<XREF:SUBM>@                           {0:M}
  +1 <<ASSOCIATION_STRUCTURE>>                    {0:M}
  +1 ALIA @<XREF:INDI>@                           {0:M}
     +2 PHRASE <Text>                             {0:1}
  +1 ANCI @<XREF:SUBM>@                           {0:M}
  +1 DESI @<XREF:SUBM>@                           {0:M}
  +1 <<IDENTIFIER_STRUCTURE>>                     {0:M}
  +1 <<CHANGE_DATE>>                              {0:1}
  +1 <<CREATION_DATE>>                            {0:1}
  +1 <<NOTE_STRUCTURE>>                           {0:M}
  +1 <<SOURCE_CITATION>>                          {0:M}
  +1 <<MULTIMEDIA_LINK>>                          {0:M}
```

The individual record is a compilation of facts or hypothesized facts about an individual. These facts may come from multiple sources. Source citations and notes allow documentation of the source where each of the facts were discovered.

A single individual may have facts distributed across multiple individual records, connected by `ALIA` (p.58) (alias, in the computing sense not the pseudonym sense) pointers. See `ALIA` for more.

Individual records are linked to Family records by use of bi-directional pointers. Details about those links are stored as substructures of the pointers in the individual record.

Other associations or relationships are represented by the `ASSO` (p.59) (association) tag. The person's relation or associate is the person being pointed to. The association or relationship is stated by the value on the subordinate `ROLE` (p.72) line.

> *Example* — This records define the `INDI` (p.66) with cross-reference identifier "`@I2@`" is the godparent of the current person:
>
> ```
> 0 @I1@ INDI
> 1 ASSO @I2@
> 2 ROLE GODP
> ```

Events stored as facts within an `INDI` record may also have `FAMC` (p.64) or `ASSO` tags to indicate families and individuals that participated in those events. For example, a `FAMC` pointer subordinate to an adoption event indicates a relationship to family by adoption; biological parents can be shown by a `FAMC` pointer subordinate to the birth event; the eulogist at a funeral can be shown by an `ASSO` pointer subordinate to the burial event; and so on.

MULTIMEDIA_RECORD :=

```
n @XREF:OBJE@ OBJE                                  {1:1}
  +1 RESN <List:Enum>                               {0:1}
  +1 FILE <Special>                                 {1:M}
     +2 FORM <Special>                              {1:1}
        +3 MEDI <Enum>                              {0:1}
           +4 PHRASE <Text>                         {0:1}
     +2 TITL <Text>                                 {0:1}
  +1 <<IDENTIFIER_STRUCTURE>>                       {0:M}
  +1 <<NOTE_STRUCTURE>>                             {0:M}
  +1 <<SOURCE_CITATION>>                            {0:M}
  +1 <<CHANGE_DATE>>                                {0:1}
  +1 <<CREATION_DATE>>                              {0:1}
```

The multimedia record refers to one ore more external digital files, and may provide some additional information about the files and the media they encode.

The file reference can occur more than once to group multiple files together. Grouped files should each pertain to the same context. For example, a sound clip and a photo both of the same event might be grouped in a single `OBJE` (p.69).

The change and creation dates should be for the `OBJE` record itself, not the underlying files.

`REPOSITORY_RECORD` :=

```
n @XREF:REPO@ REPO                                 {1:1}
  +1 NAME <Text>                                   {1:1}
  +1 <<ADDRESS_STRUCTURE>>                         {0:1}
  +1 PHON <Special>                                {0:M}
  +1 EMAIL <Special>                               {0:M}
  +1 FAX <Special>                                 {0:M}
  +1 WWW <Special>                                 {0:M}
  +1 <<NOTE_STRUCTURE>>                            {0:M}
  +1 <<IDENTIFIER_STRUCTURE>>                      {0:M}
  +1 <<CHANGE_DATE>>                               {0:1}
  +1 <<CREATION_DATE>>                             {0:1}
```

The repository record provides information about an institution or person that has a collection of sources. Informal repositories include the owner 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.

Layered repositories, such as an archive containing copies of a subset of records from another archive or archives that have moved or been bought by other archives, are not modeled in this version of GEDCOM. It is expected they will be added in a later version. Until such time, it is recommended that the repository record store current contact information, if known.

`SHARED_NOTE_RECORD` :=

```
n @XREF:SNOTE@ SNOTE <Text>                        {1:1}
  +1 MIME <Special>                                {0:1}
  +1 LANG <Special>                                {0:1}
  +1 <<SOURCE_CITATION>>                           {0:M}
  +1 <<IDENTIFIER_STRUCTURE>>                      {0:M}
  +1 <<CHANGE_DATE>>                               {0:1}
  +1 <<CREATION_DATE>>                             {0:1}
```

A catch-all location for information that does not fully fit within other structures. It may include research notes, additional context, alternative interpretations, reasoning, and so forth.

A shared note record may be pointed to by multiple other structures, and should only be used when the intended semantics is that editing the note in one place edits it in all places. If each instance of the note may be edited separately, a `NOTE` (p.69) should be used instead.

> *Example* — The origin of a name might be a reasonable shared note, while the reason a particular person was given that name may make more sense as a non-shared note.
>
> ```
> 0 @GORDON@ SNOTE "Gordon" is a traditional Scottish surname.
> 1 CONT It became a given name in honor of Charles George Gordon.
> 1 SOUR @VOID@
> 2 COMM https://en.wikipedia.org/wiki/Gordon_(given_name)
> 0 @I1@ INDI
> 1 NAME Gordon /Jones/
> 2 NOTE Named after the astronaut Gordon Cooper
> 2 SNOTE @GORDON@
> ```

> *Note* — Although sharable notes have been part of GEDCOM since version 5.0 was released in 1991, as of 2021 relatively few applications have a user interface that presents shared notes as such to users. It is recommended that `SNOTE` (p.73) be avoided when `NOTE` will suffice.

A `SHARED_NOTE_RECORD` (p.34) may contain a pointer to a `SOURCE_RECORD` and vice versa. Applications must not create GEDCOM document where these mutual pointers form a cycle. Applications should also ensure they can handle invalid files with such cycles in a safe manner.

SOURCE_RECORD :=

```
n @XREF:SOUR@ SOUR                                 {1:1}
  +1 DATA                                          {0:1}
     +2 EVEN <List:Enum>                           {0:M}
        +3 DATE <DatePeriod>                       {0:1}
           +4 PHRASE <Text>                        {0:1}
        +3 <<PLACE_STRUCTURE>>                     {0:1}
     +2 AGNC <Text>                                {0:1}
     +2 <<NOTE_STRUCTURE>>                         {0:M}
  +1 AUTH <Text>                                   {0:1}
  +1 TITL <Text>                                   {0:1}
  +1 ABBR <Text>                                   {0:1}
  +1 PUBL <Text>                                   {0:1}
  +1 TEXT <Text>                                   {0:1}
     +2 MIME <Special>                             {0:1}
     +2 LANG <Special>                             {0:1}
  +1 <<SOURCE_REPOSITORY_CITATION>>                {0:M}
  +1 <<IDENTIFIER_STRUCTURE>>                      {0:M}
  +1 <<CHANGE_DATE>>                               {0:1}
  +1 <<CREATION_DATE>>                             {0:1}
  +1 <<NOTE_STRUCTURE>>                            {0:M}
  +1 <<MULTIMEDIA_LINK>>                           {0:M}
```

A source record describes an entire source. A source *may* also point to `REPO` (p. 72)s to describe repositories or archives where the source document may be found. The part of a source relevant to a specific fact, such as a specific page or entry, is indicated in a `SOURCE_CITATION` (p.50) that points to the source record.

> *Note* — This sourcing model is known to be insufficient for some use cases and may be refined in a future version of GEDCOM.

A `SOURCE_RECORD` (p.36) may contain a pointers to a `SHARED_NOTE_RECORD` (p.34) and vice versa. Applications must not create GEDCOM document where these mutual pointers form a cycle. Applications should also ensure they can handle invalid files with such cycles in a safe manner.

`SUBMITTER_RECORD` :=

```
n @XREF:SUBM@ SUBM                               {1:1}
  +1 NAME <Text>                                 {1:1}
  +1 <<ADDRESS_STRUCTURE>>                       {0:1}
  +1 PHON <Special>                              {0:M}
  +1 EMAIL <Special>                             {0:M}
  +1 FAX <Special>                               {0:M}
  +1 WWW <Special>                               {0:M}
  +1 <<MULTIMEDIA_LINK>>                         {0:M}
  +1 LANG <Special>                              {0:M}
  +1 <<IDENTIFIER_STRUCTURE>>                    {0:M}
  +1 <<NOTE_STRUCTURE>>                          {0:M}
  +1 <<CHANGE_DATE>>                             {0:1}
  +1 <<CREATION_DATE>>                           {0:1}
```

The submitter record identifies an individual or organization that contributed information contained in the GEDCOM document. All records in the document are assumed to be contributed by the submitter referenced in the `HEAD` (p.66), unless a `SUBM` (p.74) structure inside a specific record points at a different submitter record.

3.2.3. Substructures

`ADDRESS_STRUCTURE` :=

```
n ADDR <Special>                                 {1:1}
  +1 ADR1 <Special>                              {0:1}
  +1 ADR2 <Special>                              {0:1}
  +1 ADR3 <Special>                              {0:1}
  +1 CITY <Special>                              {0:1}
  +1 STAE <Special>                              {0:1}
  +1 POST <Special>                              {0:1}
  +1 CTRY <Special>                              {0:1}
```

A specific building, plot, or location. The payload is the full formatted address as it would appear on a mailing label, including appropriate line breaks (encoded using `CONT` (p.60) tags). The expected order of address components varies by region; the address should be organized as expected by the addressed region.

Optionally, additional substructures such as `STAE` (p.74) and `CTRY` (p.60) are provided to be used by systems that have structured their addresses for indexing and sorting. If the substructures and `ADDR` (p.57) payload disagree, the `ADDR` payload shall be taken as correct. Because the regionally-correct order and formatting of address components cannot be determined from the substructures alone, the `ADDR` payload is required, even if its content appears to be redundant with the substructures.

`ASSOCIATION_STRUCTURE` :=

```
n ASSO @<XREF:INDI>@                               {1:1}
  +1 PHRASE <Text>                                 {0:1}
  +1 ROLE <Enum>                                   {1:1}
     +2 PHRASE <Text>                              {0:1}
  +1 <<SOURCE_CITATION>>                           {0:M}
  +1 <<NOTE_STRUCTURE>>                            {0:M}
```

An individual associated with the subject of the superstructure. The nature of the association is indicated in the `ROLE` (p.72) substructure.

A `voidPtr` (p.9) and `PHRASE` (p.70) can be used to describe associations to people not referenced by any `INDI` (p.66) record.

> *Example* — The following indicates that “Mr Stockdale” was the individual’s teacher and that individual `@I2@` was the clergy officiating at their baptism.

```
0 @I1@ INDI
1 ASSO @VOID@
2 PHRASE Mr Stockdale
2 ROLE OTHER
3 PHRASE Teacher
1 BAPM
2 DATE 1930
2 ASSO @I2@
3 ROLE CLERGY
```

`CHANGE_DATE` :=

```
n CHAN                                           {1:1}
  +1 DATE <DateExact>                           {1:1}
     +2 TIME <Time>                             {0:1}
  +1 <<COMMENT_STRUCTURE>>                      {0:M}
```

The date of the most recent modification of the superstructure, optionally with notes about that modification.

`COMMENT_STRUCTURE` :=

```
n COMM <Text>                                    {1:1}
  +1 MIME <Special>                             {0:1}
  +1 LANG <Special>                             {0:1}
```

Like a `NOTE_STRUCTURE` (p.47), but without a source citation substructure.

`CREATION_DATE` :=

```
n CREA                                           {1:1}
  +1 DATE <DateExact>                           {1:1}
     +2 TIME <Time>                             {0:1}
```

The date of the initial creation of the superstructure. Because this refers to the initial creation, it should not be modified after the structure is created.

`EVENT_DETAIL` :=

```
n DATE <DateValue>                                  {0:1}
  +1 TIME <Time>                                    {0:1}
  +1 PHRASE <Text>                                  {0:1}
n <<PLACE_STRUCTURE>>                               {0:1}
n <<ADDRESS_STRUCTURE>>                             {0:1}
n PHON <Special>                                    {0:M}
n EMAIL <Special>                                   {0:M}
n FAX <Special>                                     {0:M}
n WWW <Special>                                     {0:M}
n AGNC <Text>                                       {0:1}
n RELI <Text>                                       {0:1}
n CAUS <Text>                                       {0:1}
n RESN <List:Enum>                                  {0:1}
n SDATE <DateValue>                                 {0:1}
  +1 TIME <Time>                                    {0:1}
  +1 PHRASE <Text>                                  {0:1}
n <<ASSOCIATION_STRUCTURE>>                         {0:M}
n <<NOTE_STRUCTURE>>                                {0:M}
n <<SOURCE_CITATION>>                               {0:M}
n <<MULTIMEDIA_LINK>>                               {0:M}
n UID <Special>                                     {0:M}
```

Substructures that may be shared by most individual and family events and attributes.

Note that most of these substructures are limited to 1 per event. Conflicting event information should be represented by placing them in separate event structures (with appropriate source citations) rather than by placing them under the same enclosing event.

`FAMILY_ATTRIBUTE_STRUCTURE` :=

```
[
n NCHI <Integer>                                {1:1}
  +1 TYPE <Text>                                {0:1}
  +1 <<FAMILY_EVENT_DETAIL>>                    {0:1}
|
n RESI <Text>                                   {1:1}
  +1 TYPE <Text>                                {0:1}
  +1 <<FAMILY_EVENT_DETAIL>>                    {0:1}
|
n FACT <Text>                                   {1:1}
  +1 TYPE <Text>                                {1:1}
  +1 <<FAMILY_EVENT_DETAIL>>                    {0:1}
]
```

Family attributes; see Family Attributes (p.55) for more.

`FAMILY_EVENT_DETAIL` :=

```
n HUSB                                          {0:1}
  +1 AGE <Age>                                  {1:1}
     +2 PHRASE <Text>                           {0:1}
n WIFE                                          {0:1}
  +1 AGE <Age>                                  {1:1}
     +2 PHRASE <Text>                           {0:1}
n <<EVENT_DETAIL>>                              {0:1}
```

Substructures shared by most family events and attributes.

FAMILY_EVENT_STRUCTURE :=

```
[
n [ ANUL | CENS | DIV | DIVF | ENGA |
    MARB | MARC | MARL | MARS ]                  {1:1}
  +1 TYPE <Text>                                 {0:1}
  +1 <<FAMILY_EVENT_DETAIL>>                     {0:1}
|
n MARR [Y|<NULL>]                                {1:1}
  +1 TYPE <Text>                                 {0:1}
  +1 <<FAMILY_EVENT_DETAIL>>                     {0:1}
|
n EVEN <Text>                                    {1:1}
  +1 TYPE <Text>                                 {1:1}
  +1 <<FAMILY_EVENT_DETAIL>>                     {0:1}
]
```

Family events; see Family Events (p.53) for more.

For `MARR` (p.53), use `MARR Y` to indicate a marriage known to have happened but both date and place are unknown.

IDENTIFIER_STRUCTURE :=

```
[
n REFN <Special>                                 {1:1}
  +1 TYPE <Text>                                 {0:1}
|
n UID <Special>                                  {1:1}
|
n EXID <Special>                                 {1:1}
  +1 TYPE <Text>                                 {0:1}
]
```

Each of these provides an identifier for a structure or its subject, and each is different in purpose:

- `REFN` (p.72) is a user-generated identifier for a structure.
- `UID` (p.77) is a globally-unique identifier for a structure.
- `EXID` (p.63) is an identifier maintained by an external authority that applies to the subject of the structure.

`INDIVIDUAL_ATTRIBUTE_STRUCTURE` :=

```
[
n [ CAST | DSCR | EDUC | NATI | OCCU | PROP |
    RELI | RESI | TITL ] <Text>                 {1:1}
  +1 TYPE <Text>                                {0:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                {0:1}
|
n FACT <Text>                                   {1:1}
  +1 TYPE <Text>                                {1:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                {0:1}
|
n IDNO <Special>                                {1:1}
  +1 TYPE <Text>                                {1:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                {0:1}
|
n SSN <Special>                                 {1:1}
  +1 TYPE <Text>                                {0:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                {0:1}
|
n [ NCHI | NMR ] <Integer>                      {1:1}
  +1 TYPE <Text>                                {0:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                {0:1}
]
```

Individual attributes; see Individual Attributes (p.54) for more.

`INDIVIDUAL_EVENT_DETAIL` :=

```
n <<EVENT_DETAIL>>                              {1:1}
n AGE <Age>                                     {0:1}
  +1 PHRASE <Text>                              {0:1}
```

Substructures shared by most individual events and attributes.

`INDIVIDUAL_EVENT_STRUCTURE` :=

```
[
n [ BIRT | CHR ]                                          {1:1}
  +1 TYPE <Text>                                          {0:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                          {0:1}
  +1 FAMC @<XREF:FAM>@                                    {0:1}
|
n DEAT   [Y|<NULL>]                                       {1:1}
  +1 TYPE <Text>                                          {0:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                          {0:1}
|
n [ BURI | CREM | BAPM | BARM | BASM | BLES |
    CHRA | CONF | FCOM | ORDN | NATU | EMIG |
    IMMI | CENS | PROB | WILL | GRAD | RETI ]             {1:1}
  +1 TYPE <Text>                                          {0:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                          {0:1}
|
n ADOP                                                    {1:1}
  +1 TYPE <Text>                                          {0:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                          {0:1}
  +1 FAMC @<XREF:FAM>@                                    {0:1}
     +2 ADOP <Enum>                                       {0:1}
        +3 PHRASE <Text>                                  {0:1}
|
n EVEN <Text>                                             {1:1}
  +1 TYPE <Text>                                          {1:1}
  +1 <<INDIVIDUAL_EVENT_DETAIL>>                          {0:1}
]
```

Individual events; see Individual Events (p.52) for more.

For `DEAT` (p.52), use `DEAT Y` to indicate a death known to have happened but both date and place are unknown.

`LDS_INDIVIDUAL_ORDINANCE` :=

```
[
n [ BAPL | CONL | ENDL | INIT ]                {1:1}
  +1 DATE <DateValue>                          {0:1}
     +2 TIME <Time>                            {0:1}
     +2 PHRASE <Text>                          {0:1}
  +1 TEMP <Text>                               {0:1}
  +1 <<PLACE_STRUCTURE>>                       {0:1}
  +1 STAT <Enum>                               {0:1}
     +2 DATE <DateExact>                       {1:1}
        +3 TIME <Time>                         {0:1}
  +1 <<NOTE_STRUCTURE>>                        {0:M}
  +1 <<SOURCE_CITATION>>                       {0:M}
|
n SLGC                                         {1:1}
  +1 DATE <DateValue>                          {0:1}
     +2 TIME <Time>                            {0:1}
     +2 PHRASE <Text>                          {0:1}
  +1 TEMP <Text>                               {0:1}
  +1 <<PLACE_STRUCTURE>>                       {0:1}
  +1 FAMC @<XREF:FAM>@                         {1:1}
  +1 STAT <Enum>                               {0:1}
     +2 DATE <DateExact>                       {1:1}
        +3 TIME <Time>                            {0:1}
  +1 <<NOTE_STRUCTURE>>                        {0:M}
  +1 <<SOURCE_CITATION>>                       {0:M}
]
```

Ordinances performed by the Church of Jesus Christ of Latter-Day Saints; see Latter-Day Saint Ordinances (p.56) for more.

Dates for these ordinances should be in the default (`GREGORIAN` (p.88)) calendar and be 1830 or later.

LDS_SPOUSE_SEALING :=

```
n SLGS                                    {1:1}
  +1 DATE <DateValue>                     {0:1}
     +2 TIME <Time>                       {0:1}
     +2 PHRASE <Text>                     {0:1}
  +1 TEMP <Text>                          {0:1}
  +1 <<PLACE_STRUCTURE>>                  {0:1}
  +1 STAT <Enum>                          {0:1}
     +2 DATE <DateExact>                  {1:1}
        +3 TIME <Time>                    {0:1}
  +1 <<NOTE_STRUCTURE>>                   {0:M}
  +1 <<SOURCE_CITATION>>                  {0:M}
```

Ordinances performed by the Church of Jesus Christ of Latter-Day Saints; see Latter-Day Saint Ordinances (p.56) for more.

Dates for these ordinances should be in the default (`GREGORIAN` (p.88)) calendar and be 1830 or later.

MULTIMEDIA_LINK :=

```
n OBJE @<XREF:OBJE>@                      {1:1}
  +1 CROP                                 {0:1}
     +2 TOP <Integer>                     {0:1}
     +2 LEFT <Integer>                    {0:1}
     +2 HEIGHT <Integer>                  {0:1}
     +2 WIDTH <Integer>                   {0:1}
```

Links the superstructure to the `MULTIMEDIA_RECORD` (p.33) with the given pointer, optionally indicating a subregion of an image that represents or applies to the superstructure.

NON_EVENT_STRUCTURE :=

```
n NO <Enum>                               {1:1}
  +1 DATE <DatePeriod>                    {0:1}
     +2 PHRASE <Text>                     {0:1}
  +1 <<NOTE_STRUCTURE>>                   {0:M}
  +1 <<SOURCE_CITATION>>                  {0:M}
```

Indicates that an event, given in the payload, did not happen within a given date period (or never happened if there is no `DATE` (p.61) substructure).

Substructures may provide discussion about the non-occurrence of the event but must not limit the meaning of what did not occur. No substructure other than `DATE` may restrict the breadth of that negative assertion.

> *Example* — `1 NO DEAT` means "no death occurred"

> *Example* —
> ```
> 1 NO MARR
> 2 DATE TO 24 MAR 1880
> ```
> means "no marriage had occurred as of March 24th, 1880"

`NOTE_STRUCTURE` :=

```
[
n NOTE <Text>                                  {1:1}
  +1 <<SOURCE_CITATION>>                       {0:M}
  +1 MIME <Special>                            {0:1}
  +1 LANG <Special>                            {0:1}
|
n SNOTE @<XREF:SNOTE>@                         {1:1}
]
```

A catch-all location for information that does not fully fit within other structures. It may include research notes, additional context, alternative interpretations, reasoning, and so forth.

See `SHARED_NOTE_RECORD` (p.34) for advice on choosing between `NOTE` (p.69) and `SNOTE` (p.73).

`PERSONAL_NAME_PIECES` :=

```
n NPFX <Text>                                  {0:M}
n GIVN <Text>                                  {0:M}
n NICK <Text>                                  {0:M}
n SPFX <Text>                                  {0:M}
n SURN <Text>                                  {0:M}
n NSFX <Text>                                  {0:M}
```

Optional isolated name parts; see `PERSONAL_NAME_STRUCTURE` for more.

Example — "Lt. Cmndr. Joseph Allen jr." might be presented as

```
1 NAME Lt. Cmndr. Joseph /Allen/ jr.
2 NPFX Lt. Cmndr.
2 GIVN Joseph
2 SURN Allen
2 NSFX jr.
```

PERSONAL_NAME_STRUCTURE :=

```
n NAME <Personal Name>                        {1:1}
  +1 TYPE <Enum>                              {0:1}
     +2 PHRASE <Text>                         {0:1}
  +1 <<PERSONAL_NAME_PIECES>>                 {0:1}
  +1 TRAN <Personal Name>                     {0:M}
     +2 LANG <Special>                        {1:1}
     +2 <<PERSONAL_NAME_PIECES>>              {0:1}
  +1 <<NOTE_STRUCTURE>>                       {0:M}
  +1 <<SOURCE_CITATION>>                      {0:M}
```

Names of individuals are represented in the manner the name is normally spoken, with the family name, surname, or nearest cultural parallel thereunto separated by slashes (U+002F `/`). 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 `PERSONAL_NAME_PIECES` (p.47) are provided optionally for systems that cannot operate effectively with less structured information. The Personal Name payload shall be seen as the primary name representation, with name pieces as optional auxiliary information.

The name may be translated or transliterated into different languages or scripts using the `TRAN` (p.76) substructure. It is recommended, but not required, that if the name pieces are used, the same pieces are used in each translation and transliteration.

A `TYPE` (p.76) is used to specify the particular variation that this name is. For example; it could indicate that this name is a name taken at immigration or that it could be an 'also known as' name. See the `NAME`.`TYPE` enumeration (p.83) for more.

Note — Alternative approaches to representing names are being considered for future GEDCOM releases.

`PLACE_STRUCTURE` :=

```
n PLAC <List:Text>                                   {1:1}
  +1 FORM <List:Text>                                {0:1}
  +1 LANG <Special>                                  {0:1}
  +1 TRAN <List:Text>                                {0:M}
     +2 LANG <Special>                               {1:1}
  +1 MAP                                             {0:1}
     +2 LATI <Special>                               {1:1}
     +2 LONG <Special>                               {1:1}
  +1 EXID <Special>                                  {0:M}
     +2 TYPE <Text>                                  {0:1}
  +1 <<NOTE_STRUCTURE>>                              {0:M}
```

A place, which may be represented in several ways:

- The payload contains a comma-separated list of region names, ordered from smallest to largest. The specific meaning of each element is given by the `FORM` (p.65) substructure, or in the `HEAD`.`PLAC`.`FORM` if there is no `FORM` substructure. Elements should be left blank if they are unknown, do not apply to the location, or are too specific for the region in question.

  Example — A record describing births throughout Oneida county could be recorded as

  ```
  0 @S1@ SOUR
  1 DATA
  2 EVEN BIRT
  3 PLAC , Oneida, Idaho, USA
  4 FORM City, County, State, Country
  ```

- The payload may be translated or transliterated into different languages or scripts using the `TRAN` (p.76) substructure. It should use the same `FORM` as the payload.
- Global coordinates may be presented in the `MAP` (p.67) substructure

> *Note* — GEDCOM currently does not support places where a region name contains a comma. An alternative system for representing locations is likely to be added in a later version of GEDCOM.

SOURCE_CITATION :=

```
n SOUR @<XREF:SOUR>@                              {1:1}
  +1 PAGE <Text>                                  {0:1}
  +1 DATA                                         {0:1}
     +2 DATE <DateValue>                          {0:1}
        +3 TIME <Time>                            {0:1}
        +3 PHRASE <Text>                          {0:1}
     +2 TEXT <Text>                               {0:M}
        +3 MIME <Special>                         {0:1}
        +3 LANG <Special>                         {0:1}
  +1 EVEN <Enum>                                  {0:1}
     +2 PHRASE <Text>                             {0:1}
     +2 ROLE <Enum>                               {0:1}
        +3 PHRASE <Text>                          {0:1}
  +1 QUAY <Enum>                                  {0:1}
  +1 <<MULTIMEDIA_LINK>>                          {0:M}
  +1 <<COMMENT_STRUCTURE>>                        {0:M}
  +1 SNOTE @<XREF:SNOTE>@                         {0:M}
```

A citation indicating that the pointed-to source record supports the claims made in the superstructure. Substructures provide additional information about how that source applies to the subject of the citation's superstructure:

- `PAGE` (p.69): where in the source the relevant material can be found.
- `DATA` (p.61): the relevant data from the source.
- `EVEN` (p.62): what event the relevant material was recording.
- `QUAY` (p.72): an estimation of the reliability of the source in regard to these claims.
- `MULTIMEDIA_LINK` (p.46): digital copies of the cited part of the source

When no source record is available, a `voidPtr` (p.9) and accompanying `COMM` (p.59) can used to describe the source.

Example —

```
1 DSCR Tall enough his head touched the ceiling
2 SOUR @VOID@
3 COMM His grand-daughter Lydia told me this in 1980
```

`SOURCE_REPOSITORY_CITATION` :=

```
n REPO @<XREF:REPO>@                           {1:1}
  +1 <<NOTE_STRUCTURE>>                        {0:M}
  +1 CALN <Special>                            {0:M}
     +2 MEDI <Enum>                            {0:1}
        +3 PHRASE <Text>                       {0:1}
```

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 name and addresses are stored in the `REPOSITORY_RECORD` (p.34). 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 `CALN` (p.59) substructure.

3.3. Structure Meaning

3.3.1. Events

As a general rule, events are things that happen on a specific date. Use the `dateRange` (p.20) form to indicate that an event took place at some time between two dates. In most cases, a `DatePeriod` is inappropriate for an event; if the subject of your recording occurred over a period of time, then it is probably not an event, but rather an attribute.

Event structures can be used to record notes about an event without asserting the event actually occurred. An event structure asserts the event did occur if any of the following are true:

- There is a `DATE` (p.61) substructure

  Example —

  ```
  1 DEAT
  2 DATE 2 OCT 1937
  ```

- There is a `PLAC` (p.71) substructure

  Example —

  ```
  1 DEAT
  2 PLAC Cove, Cache, Utah
  ```

- The event has a payload. A special payload `Y` can be used with some event types to indicate that the event is known to have occurred without providing any additional information about it.

  Example —
  ```
  1 DEAT Y
  ```

If none of the above are true, the structure should be seen as a place for inconclusive research notes about the possibility of the event. An assertion that an event did not occur should be encoded using the `NO` (p.68) structure.

3.3.1.1. Individual Events

Tag	Name	Description
`ADOP`	adoption	Creation of a legally approved child-parent relationship that does not exist biologically.
`BAPM`	baptism	Baptism, performed in infancy or later. (See also `BAPL` (p.56) and `CHR`.)
`BARM`	Bar Mitzvah	The ceremonial event held when a Jewish boy reaches age 13.
`BASM`	Bas Mitzvah	The ceremonial event held when a Jewish girl reaches age 13, also known as “Bat Mitzvah.”
`BIRT`	birth	Entering into life.
`BLES`	blessing	Bestowing divine care or intercession. Sometimes given in connection with a naming ceremony.
`BURI`	burial	Disposing of the mortal remains of a deceased person.
`CENS`	census	Periodic count of the population for a designated locality, such as a national or state Census.
`CHR`	christening	Baptism or naming events for a child.
`CHRA`	adult christening	Baptism or naming events for an adult person.
`CONF`	confirmation	Conferring full church membership.
`CREM`	cremation	Disposal of the remains of a person’s body by fire.

Tag	Name	Description
`DEAT`	death	Mortal life terminates.
`EMIG`	emigration	Leaving one's homeland with the intent of residing elsewhere.
`FCOM`	first communion	The first act of sharing in the Lord's supper as part of church worship.
`GRAD`	graduation	Awarding educational diplomas or degrees to individuals.
`IMMI`	immigration	Entering into a new locality with the intent of residing there.
`NATU`	naturalization	Obtaining citizenship.
`ORDN`	ordination	Receiving authority to act in religious matters.
`PROB`	probate	Judicial determination of the validity of a will. It may indicate several related court activities over several dates.
`RETI`	retirement	Exiting an occupational relationship with an employer after a qualifying time period.
`WILL`	will	A legal document treated as an event, by which a person disposes of his or her estate. It takes effect after death. The event date is the date the will was signed while the person was alive. (See also `PROB`)

In addition, `EVEN` (p.62) is a structure for a generic individual event. It must have a `TYPE` (p.76) substructure to define what kind of event is being provided.

3.3.1.2. Family Events

Tag	Name	Description
`ANUL`	annulment	Declaring a marriage void from the beginning (never existed).
`CENS`	census	Periodic count of the population for a designated locality, such as a national or state Census.
`DIV`	divorce	Dissolving a marriage through civil action.
`DIVF`	divorce filed	Filing for a divorce by a spouse.

Tag	Name	Description
`ENGA`	engagement	Recording or announcing an agreement between 2 people to become married.
`MARB`	marriage bann	Official public notice given that 2 people intend to marry.
`MARC`	marriage contract	Recording a formal agreement of marriage, including the prenuptial agreement in which marriage partners reach agreement about the property rights of 1 or both, securing property to their children.
`MARL`	marriage license	Obtaining a legal license to marry.
`MARR`	marriage	A legal, common-law, or customary event that creates a family unit of a man and a woman as husband and wife.
`MARS`	marriage settlement	Creating an agreement between 2 people contemplating marriage, at which time they agree to release or modify property rights that would otherwise arise from the marriage.

In addition, `EVEN` is a structure for a generic family event. It must have a `TYPE` substructure to define what kind of event is being provided.

3.3.2. Attributes

Unlike events, the presence of an attribute is sufficient to assert the attribute applied to the individual, regardless of the attribute's substructures and payload.

3.3.2.1. Individual Attributes

Tag	Name	Description
`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, or occupation.
`DSCR`	physical description	The physical characteristics of a person, place, or thing.

Tag	Name	Description
`EDUC`	education	Indicator of a level of education attained.
`IDNO`	identifying number	A number or other string assigned to identify a person within some significant external system. It must have a `TYPE` substructure to define what kind of identification number is being provided.
`NATI`	nationality	The national heritage of an individual.
`NCHI`	number of children	The number of children that this person is known to be the parent of (all marriages).
`NMR`	number of marriages	The number of times this person has participated in a family as a spouse or parent.
`OCCU`	occupation	The type of work or profession of an individual.
`PROP`	property	Pertaining to possessions such as real estate or other property of interest.
`RELI`	religion	A religious denomination to which a person is affiliated or for which a record applies.
`RESI`	residence	An address or place of residence where an individual resided.
`SSN`	social security number	A number assigned by the United States Social Security Administration, used for tax identification purposes. It is a type of `IDNO`.
`TITL`	title	A formal designation used by an individual in connection with positions of royalty or other social status, such as Grand Duke.

In addition, `FACT` (p.64) is a structure for a generic individual attribute. It must have a `TYPE` (p.76) substructure to define what kind of attribute is being provided.

3.3.2.2. Family Attributes

Tag	Name	Description
`NCHI`	number of children	The number of children that belong to this family.

Tag	Name	Description
`RESI`	residence	An address or place of residence where a family resided.

In addition, `FACT` is a structure for a generic family attribute. It must have a `TYPE` substructure to define what kind of attribute is being provided.

3.3.3. Latter-Day Saint Ordinances

The structures describing ordinances performed by the Church of Jesus Christ of Latter-Day Saints are unlike regular events in that they might either be performed during life or by proxy on the behalf of a deceased individual.

Proxy ordinances on behalf of deceased persons were once requested and officially recorded using GEDCOM. This is no longer the case; however, it should be noted that when it was the case the following two principles held:

- `PLAC` (p.71) was used only for ordinances that were performed by the recipient in life.
- `TEMP` (p.75) was used with all `ENDL`, `SLGC`, and `SLGS`, but only with posthumous proxy `BAPL` and `CONL`.

Tag	Name	Description
`BAPL`	baptism	The event of baptism performed at age eight or later by priesthood authority of The Church of Jesus Christ of Latter-day Saints. (See also `BAPM` (p.52))
`CONL`	confirmation	The religious event by which a person receives membership in The Church of Jesus Christ of Latter-day Saints. (See also `CONF` (p.52))
`INIT`	initiatory	A religious event where an initiatory ordinance for an individual was performed by priesthood authority in a temple of The Church of Jesus Christ of Latter-day Saints.
`ENDL`	endowment	A religious event where an endowment ordinance for an individual was performed by priesthood authority in a temple of The Church of Jesus Christ of Latter-day Saints.

Tag	Name	Description
`SLGC`	sealing child	A religious event pertaining to the sealing of a child to his or her parents in a temple ceremony of The Church of Jesus Christ of Latter-day Saints.
`SLGS`	sealing spouse	A religious event pertaining to the sealing of a husband and wife in a temple ceremony of The Church of Jesus Christ of Latter-day Saints. (See also `MARR` (p.53))

3.3.4. Other structure types

The type of a structure can generally be determined by its tag, and are listed in this section alphabetically by tag. Some tags have special meaning in certain contexts; those context-defined meanings cannot be used with that tag in extensions.

`ABBR` (Abbreviation)

A short name of a title, description, or name used for sorting, filing, and retrieving records.

`ADOP` (Adoption)

An [Individual Event].

The `ADOP` tag is also used in a context-defined way for one other structure:

`FAMC.ADOP`

An enumerated value (p.79) indicating which parent(s) in the family adopted this individual.

`ADDR` (Address)

The location of, or most relevant to, the subject of the superstructure. See `ADDRESS_STRUCTURE` (p.37) for more.

`ADR1` (Address Line 1)

The first line of the address, used for indexing. This is the value of the line corresponding to the `ADDR` tag line in the address structure. See `ADDRESS_STRUCTURE` for more.

`ADR2` (Address Line 2)

The second line of the address, used for indexing. This is the value of the first `CONT` (p.60) line subordinate to the `ADDR` tag in the address structure. See `ADDRESS_STRUCTURE` for more.

`ADR3` (Address Line 3)

The third line of the address, used for indexing. This is the value of the second `CONT` line subordinate to the `ADDR` tag in the address structure. See `ADDRESS_STRUCTURE` for more.

`AGE` (Age at event)

The age of the individual at the time an event occurred, or the age listed in the document.

`AGNC` (Responsible agency)

The organization, institution, corporation, person, or other entity that has responsibility for the associated context. Examples are an employer of a person of an associated occupation, or a Church that administered rites or events, or an organization responsible for creating or archiving records.

`ALIA` (Alias)

A single individual may have facts distributed across multiple individual records, connected by `ALIA` pointers (named after "alias" in the computing sense, not the pseudonym sense).

> *Note* — This standard does not define how to connect `INDI` (p.66) records with `ALIA`. Some systems organize `ALIA` pointers to create a tree structure, with the root `INDI` record containing the composite view of all facts in the leaf `INDI` records. Others distribute events and attributes between `INDI` records mutually linked by symmetric pairs of `ALIA` pointers. A future version of GEDCOM may adjust the definition of `ALIA`.

`ANCI` (Ancestor interest)

Indicates an interest in additional research for ancestors of this individual. (See also `DESI` (p.62)).

`ASSO` (Associates)

A pointer to an associated individual. See `ASSOCIATION_STRUCTURE` (p.38) for more.

`AUTH` (Author)

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.

`CALN` (Call number)

An identification or reference description used to file and retrieve items from the holdings of a repository. Despite the word "number" in the name, may contain any character, not just digits.

`CAUS` (Cause)

The reasons which precipitated an event. It is often used subordinate to a death event to show cause of death, such as might be listed on a death certificate.

`CHAN` (Change)

The most recent change to the superstructure. This is metadata about the structure itself, not data about its subject. See `CHANGE_DATE` (p.39) for more.

`CHIL` (Child)

The child in a family, whether biological, adopted, foster, sealed, or other relationship.

`CITY` (City)

The name of the city used in the address. See `ADDRESS_STRUCTURE` (p.37) for more.

`COMM` (Comment)

A `COMMENT_STRUCTURE` (p.39), which has the same semantic meaning as a `NOTE_STRUCTURE` (p.47) but may not have `SOUR` (p.74) substructures.

The `COMM` tag is also used in a context-defined way for one other structure:

`HEAD`.`COMM`

A note that a user enters to describe the contents of the document in terms of "ancestors or descendants of" so that the person receiving the data knows what genealogical information the document contains.

`CONT` (Continued)

A pseudo-structure to indicate a line break. See Lines (p.9) for more.

`COPR` (Copyright)

A copyright statement, as appropriate for the copyright laws applicable to this data.

`CORP` (Corporate name)

The name of the business, corporation, or person that produced or commissioned the product.

`CREA` (Creation)

The initial creation of the superstructure. This is metadata about the structure itself, not data about its subject. See `CREATION_DATE` (p.39) for more.

`CROP`

A subregion of an image to display. It is only valid when the superstructure links to an image with a defined pixel unit.

`LEFT` (p.67) and `TOP` (p.76) indicate the top-left corner of the region to display. `WIDTH` (p.78) and `HEIGHT` (p.66) indicate how many pixels wide and tall the region to display is. If omitted, `LEFT` and `TOP` each default to 0; `WIDTH` defaults to the image width minus `LEFT`; and `HEIGHT` defaults to the image height minus `TOP`.

The following are errors:

- `LEFT` or `LEFT` + `WIDTH` exceed the image width.
- `TOP` or `TOP` + `HEIGHT` exceed the image height.
- `CROP` applied to a non-image or image without a defined pixel unit.

`CTRY` (Country)

The name of the country that pertains to the associated address. See `ADDRESS_STRUCTURE` (p.37) for more.

`DATA`

A structure with no payload used to distinguish a description of something from metadata about it. For example, `SOUR` (p.74) and its other substructures describe a source itself, while `SOUR`.`DATA` describes the content of the source.

The `DATA` tag is also used in a context-defined way for one other structure:

`HEAD`.`SOUR`.`DATA` (Source Data)

The electronic data source or digital repository from which this GEDCOM document was exported. The payload is the name of that source, with substructures providing additional details.

`DATE`

The principle date of the subject of the superstructure. The payload is a `DateValue` (p.20), and the `DATE` structure may have a `PHRASE` (p.70) substructure.

The `DATE` tag is also used in a context-defined way for several other structures:

`CHAN`.`DATE`

The `DateExact` that the superstructure was last modified.

`CREA`.`DATE`

The `DateExact` that the superstructure was first created.

`HEAD`.`DATE`

The `DateExact` that this document was created.

`HEAD`.`SOUR`.`DATA`.`DATE`

The publication `DateExact` of the source from which this document was extracted.

(Latter-Day Saint Ordinance).`DATE`

The `DateValue` when the ordinance was performed. Latter-day Saint Ordinances (p.56) may be performed by proxy after an individual's death. The presence of a temple ordinance date should not be taken to suggest the individual was living at that time.

(Latter-Day Saint Ordinance).`STAT`.`DATE`

The `DateExact` when the status of ordinance was last updated.

`NO.DATE`

The `DatePeriod` during which the event did not occur or the attribute did not apply.

`SOUR.DATA.EVENT.DATE`

The `DatePeriod` covered by the entire source; the period during which this source recorded events.

`DESI` (Descendant Interest)

Indicates an interest in research to identify additional descendants of this individual. See also `ANCI` (p.58).

`DEST` (Destination)

An identifier for the system expected to receive this document. See `HEAD.SOUR` for guidance on choosing identifiers.

`EMAIL` (Email)

An electronic mail address, as defined by any relevant standard such as RFC 3696, RFC 5321, or RFC 5322.

If an invalid email address is present upon import, it should be preserved as-is on export.

> *Note* — GEDCOM 5.5.1 contained a typo where this tag was sometimes written `EMAI` and sometimes written `EMAIL`. `EMAIL` should be used in GEDCOM 7.0 and later.

`EVEN` (Event)

An event: a noteworthy happening related to an individual or family. If a specific event type exists, it should be used instead of a generic `EVEN` structure. Each `EVEN` must be classified by a subordinate use of the `TYPE` (p.76) tag and may be further described in the structure's payload.

Example — A person that signed a lease for land dated October 2, 1837 and a lease for mining equipment dated November 4, 1837 would be written in GEDCOM as:

```
0 @I1@ INDI
1 EVEN
2 TYPE Land Lease
2 DATE 2 OCT 1837
1 EVEN Mining equipment
2 TYPE Equipment Lease
2 DATE 4 NOV 1837
```

The `EVEN` tag is also used in a context-defined way for two other structures:

`DATA`.`EVEN`

A list of enumerated values (p.79) indicating the types of events that were recorded in a particular source. Each event type is separated by a comma and space. For example, a parish register of births, deaths, and marriages would be `BIRT, DEAT, MARR`.

`SOUR`.`EVEN`

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` (p.52) regardless of the assertions made from that record, such as the mother's name or mother's birth date.

`EXID` (External Identifier)

An identifier for the subject of the superstructure. The identifier is maintained by some external authority; the authority owning the identifier is provided in the TYPE substructure; see `EXID`.`TYPE` for more.

Depending on the maintaining authority, an `EXID` may be a unique identifier for the subject, an identifier for one of several views of the subject, or an identifier for the externally-maintained copy of the same information as is contained in this structure. However, unlike `UID` (p.77) and `REFN` (p.72), `EXID` does not identify a GEDCOM structure and structures with the same `EXID` may have originated independently rather than by edits from the same starting point.

`FAM` (Family record)

See `FAMILY_RECORD` (p.30)

`FACT`

A noteworthy attribute or fact concerning an individual or family. If a specific attribute type exists, it should be used instead of a generic `FACT` structure. Each `FACT` must be classified by a subordinate use of the `TYPE` (p.76) tag and may be further described in the structure's payload.

> *Example* — If the attribute being defined was one of the person's skills, such as woodworking, the `FACT` tag would have the value of "Woodworking", followed by a subordinate `TYPE` tag with the value "Skills".
>
> ```
> 0 @I1@ INDI
> 1 FACT Woodworking
> 2 TYPE Skills
> ```

`FAMC` (Family child)

The family in which an individual appears as a child. It is also used with a `STAT` (p.74) substructure to show individuals who are not children of the family. See `FAM` (p.63) and `FAMC`.`STAT` for more.

`FAMS` (Family spouse)

The family in which an individual appears as a partner. See `FAM` for more.

`FAX` (Facsimile)

A fax telephone number appropriate for sending data facsimiles. See `PHON` (p.70) for more.

`FILE`

A reference to an external file. Syntactically, the payload is a URL, as defined by IETF RFC 1808 and https://url.spec.whatwg.org/. However, only some URLs may be used:

- A URL with scheme `ftp`, `http`, or `https` refers to a **web-accessible file**.
- A URL with scheme `file` refers to a **machine-local file**. Machine-local files must not be used in GEDZIP (p.85) nor when sharing GEDCOM documents on the web or with unknown parties, but may be used for close collaboration between parties with known similar file structures.
- A URL with all of the following:
  - no scheme

- not beginning with `/` (U+002F)
- not containing any path segments equal to `..` (U+002E U+002E)
- not containing a reverse solidus character (U+005C `\`) or `banned` (p.7) character, either directly or in escaped form
- no query or fragment

refers to a **local file**. If the GEDCOM document is part of a GEDZIP (p.85), the URL of the local file is a zip archive filename; otherwise, the URL of a local file is resolved with *base* equal to the directory containing the GEDCOM file.

The meaning of a `FILE` (p.64) payload with any URL format not listed above is undefined in GEDCOM 7.0, but may be defined in subsequent versions of GEDCOM.

`FORM` (Format)

The media type of the file referenced by the superstructure. This should be a valid media type as defined by IETC BCP 13. A registry of media types is maintained publicly by the IANA.

The `FORM` tag is also used in a context-defined way for two other structures:

`HEAD`.`PLAC`.`FORM` (Default place format)

Any `PLAC` (p.71) with no `FORM` shall be treated as if it has this `FORM`.

`PLAC`.`FORM` (Place format)

A comma-separated list of jurisdictional titles, which has the same number of elements and in the same order as the `PLAC` structure. Entries should be in an order where each is typically subsumed by the next.

> *Example* — The following represents Baltimore, a city that is not within a county.
>
> ```
> 2 PLAC Baltimore, , Maryland, USA
> 3 FORM City, County, State, Country
> ```

`GEDC` (Gedcom)

A container for information about the entire document.

It is recommended that applications write `GEDC` with its required subrecord `VERS` (p.78) as the first substructure under `HEAD`.

`GIVN` (Given name)

A given or earned name used for official identification of a person.

`HEAD` (Header)

A pseudo-structure for storing metadata about the document. See The Header and Trailer (p.13) for more.

`HEIGHT`

How many pixels to display vertically for the image. See `CROP` (p.60) for more.

> *Note* — `HEIGHT` is a number of pixels. The correct tag for the height of an individual is the `DSCR` (p.54) attribute.
>
> > *Example* —
> > ```
> > 0 @I45@ INDI
> > 1 DSCR brown eyes, 5ft 10in, 198 pounds
> > ```

`HUSB` (Husband)

A container for information relevant to the subject of the superstructure specific to the individual described by the associated `FAM` (p.63)'s `HUSB` substructure.

The `HUSB` tag is also used in a context-defined way for one other structure:

`FAM.HUSB`

This is a partner in a `FAM` record. See `FAMILY_RECORD` (p.30) for more.

`INDI` (Individual)

See `INDIVIDUAL_RECORD` (p.32).

`LANG` (Language)

The human language of the superstructure. In the header, this provides the default language for the entire document. In a `SUBMITTER_RECORD` (p.37), it provides a language the subject of that record understands. Elsewhere it provides the language in which the `Text`-typed payloads of superstructure and its substructures appears.

The payload of `LANG` structure is a language tag, as defined by IETF BCP 47. A registry of component subtags is maintained publicly by the IANA.

`LATI` (Latitude)

A latitudinal coordinate. The payload is either `N` (for a coordinate north of the equator) or `S` (for a coordinate south of the equator) followed by a decimal number of degrees. Minutes and seconds are not used and should be converted to fractional degrees prior to encoding.

> *Example* — 18 degrees, 9 minutes, and 3.4 seconds North would be formatted as `N18.150944`.

`LEFT`

Left is a number of pixels to not display from the left side of the image. See `CROP` (p.60) for more.

`LONG` (Longitude)

A longitudinal coordinate. The payload is either `E` (for a coordinate east of the prime meridian) or `W` (for a coordinate west of the prime meridian) followed by a decimal number of degrees. Minutes and seconds are not used and should be converted to fractional degrees prior to encoding.

> *Example* — 168 degrees, 9 minutes, and 3.4 seconds East would be formatted as `E168.150944`.

`MAP` (Map)

A representative point for a location, as defined by `LATI` (p.67) and `LONG` sub-structures.

Note that `MAP` provides neither a notion of accuracy (for example, the `MAP` for a birth event may be some distance from the point where the birth occurred) nor a notion of region size (for example, the `MAP` for a place "Belarus" may be any-where within that nation's 200,000 square kilometer area).

`MEDI` (Medium)

An enumerated value (p.79) providing information about the media or the medi-um in which information is stored.

`MIME` (Media type)

Indicates the media type of the payload of the superstructure, as defined by IETC BCP 13.

Only 2 media types are supported by this structure in this version of GEDCOM:

- `text/plain` shall be presented to the user as-is, preserving all spacing, line breaks, and so forth.
- `text/html` uses HTML tags to provide presentation information. Applications should support at least the following:
  - `p` and `br` elements for paragraphing and line breaks.
  - `b`, `i`, and `u` elements for bold, italic, and underlined text (or corresponding display in other locales; see HTML §4.5 for more).
  - `sup` and `sub` elements for super- and sub-script.
  - The 3 XML entities that appear in text: `&amp;`, `&lt;` `&gt;`. Note that `&quote;` and `&apos;` are only needed in attributes. Other entities should be represented as their respective Unicode characters instead.

  Supporting more of HTML is encouraged. Unsupported elements should be ignored during display.

> *Note* — Media types are also used by external files, as described under `FORM` (p.65). External file media types are not limited to `text/plain` and `text/html`.

`NAME`

The name of the superstructure's subject, represented as a simple string.

The `NAME` tag is also used in a context-defined way for one other structure:

`INDI`.`NAME`

A `PERSONAL_NAME_STRUCTURE` (p.48) with parts, translations, sources, and so forth.

`NICK` (Nickname)

A descriptive or familiar name that is used instead of, or in addition to, one's proper name. Should only be used when a nickname is commonly written as part of a full name; for stand-alone nicknames, a separate `NAME` with `TYPE NICK` is more appropriate.

`NO` (Did not happen)

Identifies an event type which did not occur to the superstructure's subject. See `NON_EVENT_STRUCTURE` (p.46) for more.

The payload is a single event-type tag name. See Events (p.51).

`NOTE`

A `NOTE_STRUCTURE` (p.47), containing additional information provided by the submitter for understanding the enclosing data.

The `NOTE` tag is also used in a context-defined way for one other structure:

`CHAN.NOTE`

Optional notes about the change made to the superstructure. May be describe previous changes as well as the most recent, although only the most recent is described by the `CHAN`.`DATE`.

`NPFX` (Name prefix)

Text which appears on a name line before the given and surname parts of a name.

`NSFX` (Name suffix)

Text which appears on a name line after or behind the given and surname parts of a name.

`OBJE` (Object)

An external file containing information pertinent to the subject of the superstructure if a substructure, or a description of such a file if a record. See `MULTIMEDIA_RECORD` (p.33) and `MULTIMEDIA_LINK` (p.46) for more.

`PAGE` (Page)

A specific location within the information referenced. For a published work, this could include the volume of a multi-volume work and the page number or numbers. For a periodical, it could include volume, issue, and page numbers. For a newspaper, it could include a date, page number, and column number. For an unpublished source or microfilmed works, this could be a film or sheet number, page number, or frame number. A census record might have an enumerating district, page number, line number, dwelling number, and family number.

It is recommended that the data in this field be formatted comma-separated with label: value pairs

Example —

```
2 SOUR @S1@
3 PAGE Film: 1234567, Frame: 344, Line: 28
```

`PEDI` (Pedigree)

An enumerated value (p.80) indicating the type of child-to-family relationship represented by the superstructure.

`PHON` (Phone)

A telephone number. Telephone numbers have many regional variations and may contain non-digit characters. Users should be encouraged to use internationalized telephone numbers rather than local versions. As a starting point for this recommendation, there are international standards that use a '+' shorthand for the international prefix (for example, in place of 011 in the US or 00 in the UK). Examples are `+1 (555) 555-1234` (US) or `+44 20 1234 1234` (UK).

See ITU standards E.123 and E.164 for more information.

`PHRASE`

Textual information that cannot be expressed in the superstructure due to the limitations of its datatype.

> *Example* — A date interpreted from the phrase "The Feast of St John" might be
>
> ```
> 2 DATE 24 JUNE 1852
> 3 PHRASE During the feast of St John
> ```

> *Example* — A record using `1648/9` to indicate a change in new year might become
>
> ```
> 2 DATE 30 JAN 1649
> 3 PHRASE 30th of January, 1648/9
> ```

> *Example* — A record using `1648/9` to indicate uncertainty in the year might become
>
> ```
> 2 DATE BET 1648 AND 1649
> 3 PHRASE 1648/9
> ```

> *Example* — A record using `Q1 1867` to indicate an event occurred sometime within the first quarter of 1867 might become
>
> ```
> 2 DATE BET 1 JAN 1867 AND 31 MAR 1867
> 3 PHRASE Q1 1867
> ```

Example — A record defining the Maid of Honor in a marriage might become

```
1 MARR
2 ASSO @I2@
3 ROLE OTHER
4 PHRASE Maid of Honor
```

Example — A name given to a foundling orphan might be

```
1 NAME Mary //
2 GIVN Mary
2 TYPE OTHER
3 PHRASE given by orphanage
```

`PLAC` (Place)

The principle place in which the superstructure's subject occurred.

The `PLAC` tag is also used in a context-defined way for one other structure:

`HEAD`.`PLAC`

This is a placeholder for providing a default `PLAC`.`FORM`, and must not have a payload.

`POST` (Postal code)

A code used by a postal service to identify an area to facilitate mail handling. See `ADDRESS_STRUCTURE` (p.37) for more.

`PUBL` (Publication)

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.

`QUAY` (Quality of data)

An enumerated value (p.80) indicating the credibility of a piece of information, based on 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 receivers' need to evaluate the evidence for themselves.

`REFN` (Reference)

A user-defined number or text that the submitter uses to identify the superstructure. 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.

This is metadata about the structure itself, not data about its subject. Multiple structures describing different aspects of the same subject would have different `REFN` values.

`REPO` (Repository)

For a record, see `REPOSITORY_RECORD` (p.34) . For a substructure, see `SOURCE_REPOSITORY_CITATION` (p.51).

`RESN` (Restriction)

An List (p.23) of enumerated value (p.81)s signifying access to information may be denied or otherwise restricted.

The `RESN` structure is provided to assist software in filtering data that should not be exported or otherwise used in a particular context. It is recommended that tools provide an interface to allow users to filter data on export such that certain `RESN` structure payload entries result in the `RESN` structure and its superstructure being removed from the export. Such removal must not violate GEDCOM constraints: see Removing data (p.18) for more.

This is metadata about the structure itself, not data about its subject.

`ROLE` (Role)

An enumerated value (p.81) indicating what role this person played in an event or person's life.

Example — The following indicates a child's birth record as the source of the mother's name:

```
0 @I1@ INDI
1 NAME Mary //
2 SOUR @S1@
3 EVEN BIRT
4 ROLE MOTH
```

Example — The following indicates that a person's best friend was a witness at their baptism:

```
0 @I2@ INDI
1 ASSO @I3@
2 ROLE FRIEND
3 PHRASE best friend
1 BAPM
2 ASSO @I3@
3 ROLE WITN
```

`SCHMA` (Extension schema)

A container for storing meta-information about the extension tags used in this document. See Extensions (p.13) for more.

`SDATE` (Sort date)

A date to be used as a sorting hint. It is intended for use when the actual date is unknown, but the display order may be dependent on date.

If both a `DATE` (p.61) and `SDATE` are present in the same structure, the `SDATE` should be used for sorting and positioning while the `DATE` should be displayed as the date of the structure.

`SEX` (Sex)

An enumerated value (p.82) that indicates the sex of the individual at birth.

`SNOTE` (Shared Note)

A note that is shared by multiple structures (if a record) or a pointer to such a note (if a substructure). See `SHARED_NOTE_RECORD` (p.34) and `NOTE_STRUCTURE` (p. 47) for more.

`SOUR` (Source)

A description of an entire source (if a record) or the relevant part thereof to support the containing data (if a substructure). See `SOURCE_RECORD` (p.36) and `SOURCE_CITATION` (p.50) for more.

The `SOUR` tag is also used in a context-defined way for one other structure:

`HEAD`.`SOUR` (Source System)

An identifier for the product producing this GEDCOM document. A registration process for these identifiers existed for a time, but no longer does. If an existing identifier is known, it should be used. Otherwise, a URI owned by the product should be used instead.

`SPFX` (Surname prefix)

A name piece used as a non-indexing pre-part of a surname.

`SURN` (Surname)

A family name passed on or used by members of a family.

`STAE` (State)

A geographical division of a larger jurisdictional area, such as a State within the United States of America. See `ADDRESS_STRUCTURE` (p.37) for more.

`STAT` (Status)

An enumerated value assessing of the state or condition of something. Expected values differ by context; see `FAMC`.`STAT` (p.82) and (Latter-Day Saint Ordinance).`STAT` (p.83) for more.

`SUBM` (Submitter)

A contributor of information in the substructure, or a description of such a contributor if a record. As a substructure, this is metadata about the structure itself, not data about its subject. See `SUBMITTER_RECORD` (p.37) for more.

`TAG` (Extension tag)

Information relating to a single extension tag as used in this document. See Extensions (p.13) for more.

`TEMP` (Temple)

The name of a temple of The Church of Jesus Christ of Latter-day Saints. Previous editions of GEDCOM recommended using a set of abbreviations for temple names, but the list of abbreviations is no longer published by the Church and using abbreviations is no longer recommended.

`TEXT` (Text from Source)

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.

`TIME` (Time)

A time value in a 24-hour clock format.

`TITL` (Title)

The title, formal or informal, of the superstructure.

A published work, such as a book, 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, including most digital files, titles should be descriptive and appropriate to the work.

> *Example —*
>
> - The `TITL` of a letter might include the date, the sender, and the receiver.
> - The `TITL` of a transaction between a buyer and seller might have their names and the transaction date.
> - The `TITL` of a family Bible containing genealogical information might have past and present owners and a physical description of the book.
> - The `TITL` of a personal interview would cite the informant and interviewer.

The `TITL` tag is also used in a context-defined way for one other structure:

`INDI`.`TITL`

An Individual Attribute (p.54)

`TOP`

A number of pixels to not display from the top side of the image. See `CROP` (p.60) for more.

`TRLR` (Trailer)

A pseudo-structure marking the end of a GEDCOM document. See The Header and Trailer (p.13) for more.

`TRAN` (Translation or transliteration)

A translation or transliteration of the superstructure. It presents information in the same format as the superstructure, but with a different language tag. See `PLACE_STRUCTURE` (p.49), `PERSONAL_NAME_STRUCTURE` (p.48), and `LANG` (p.66) for more.

`TYPE`

A descriptive word or phrase used to further classify the superstructure.

When both a `NOTE` (p.69) and free-text `TYPE` are permitted as substructures of the same structure, the displaying systems should always display the `TYPE` value when they display the data from the associated structure; `NOTE` will typically be visible only in a detailed view.

`TYPE` must be used whenever the generic `EVEN` (p.62), `FACT` (p.64) and `IDNO` (p.54) tags are used. It may also be used for any other event or attribute.

Using the subordinate `TYPE` classification method provides a further classification of the superstructure but does not change its basic meaning.

Example — A MARR (p.53) with a TYPE could clarify what kind of marriage was performed:

```
0 @I1@ INDI
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 (p.52) (birth)
- "Tribal Custom" as a qualifier to MARR (marriage)
- "College" as a qualifier to GRAD (graduation)

See also FACT (p.64) and EVEN (p.62) for additional examples.

The TYPE tag is also used in a context-defined way for two other structures:

NAME.TYPE

An enumerated value (p.83) indicating the type of the name thereof.

EXID.TYPE

The authority issuing the EXID (p.63), represented as a URI. It is recommended that this be a URL.

If the authority maintains stable URLs for each identifier it issues, it is recommended that the TYPE payload be selected such that appending the EXID payload to it yields that URL. However, this is not required and a different URI for the set of issued identifiers may be used instead.

UID (Unique Identifier)

A globally-unique identifier of the superstructure, to be preserved across edits. If a globally-unique identifier for the record already exists, it should be used without modification, not even whitespace or letter case normalization. New globally unique identifiers should be created and formatted as described in RFC 4122.

This is metadata about the structure itself, not data about its subject. Multiple structures describing different aspects of the same subject would have different UID values.

> *Note* — Some systems used a 16-byte UUID with a custom 2-byte checksum for a total of 18 bytes:
>
> - checksum byte 1 = (sum of (byte$_i$) for i 1 through 16) mod 256
> - checksum byte 2 = (sum of ((16 – i) × (byte$_i$)) for i 1 through 16) mod 256
>
> Use of checksums for UIDs is discouraged except in cases where error-prone input is expected and an appropriate action to take in case of an error is known.

`VERS` (Version)

An identifier that represents the version level assigned to the associated product. It is defined and changed by the creators of the product.

The `VERS` tag is also used in a context-defined way for one other structure:

`GEDC`.`VERS`

The official GEDCOM standard version number. See A Guide to Version Numbers (p.5) for more.

`WIDTH`

How many pixels to display horizontally for the image. See `CROP` (p.60) for more.

`WIFE`

A container for information relevant to the subject of the superstructure specific to the individual described by the associated `FAM` (p.63)’s `WIFE` substructure.

The `WIFE` tag is also used in a context-defined way for one other structure:

`FAM`.`WIFE`

A partner in a `FAM` record. See `FAMILY_RECORD` (p.30) for more.

`WWW` (Web address)

A URL or other locator for a World Wide Web page, as defined by any relevant standard such as whatwg/url, RFC 3986, RFC 3987, and so forth.

If an invalid web address is present upon import, it should be preserved as-is on export.

3.4. Enumeration Values

`FAMC`.`ADOP`

Value	Meaning
`HUSB`	Adopted by the `HUSB` of the `FAM` pointed to by `FAMC`
`WIFE`	Adopted by the `WIFE` of the `FAM` pointed to by `FAMC`
`BOTH`	Adopted by both `HUSB` and `WIFE` of the `FAM` pointed to by `FAMC`

`DATA`.`EVEN`

A comma-separated list of event- and attribute-type tag names. See Events (p.51) and Attributes (p.54).

`SOUR`.`EVEN`

An event-type tag name. See Events (p.51) and `SOUR.EVEN` (p.63).

`MEDI`

Value	Meaning
`AUDIO`	An audio recording
`BOOK`	A bound book
`CARD`	A card or file entry
`ELECTRONIC`	A digital artifact
`FICHE`	Microfiche
`FILM`	Microfilm
`MAGAZINE`	Printed periodical
`MANUSCRIPT`	Written pages
`MAP`	Cartographic map
`NEWSPAPER`	Printed newspaper
`PHOTO`	Photograph
`TOMBSTONE`	Burial marker or related memorial

Value	Meaning
`VIDEO`	Motion picture recording
`OTHER`	A medium not listed here; should have a `PHRASE` substructure

`PEDI`

Value	Meaning
`ADOPTED`	adoptive parents.
`BIRTH`	birth parents.
`FOSTER`	the child was included in a foster or guardian family.
`SEALING`	the child was sealed to parents other than birth parents.
`OTHER`	A connection not listed here; should have a `PHRASE` substructure

> *Note* — The structures for foster children in particular, and family relationships in general, are known to have undesirable limitations and are likely to change in a future version of GEDCOM.

`NO`

A single event- or attribute-type tag name. See Events (p.51) and Attributes (p.54).

`QUAY`

Value	Meaning
`0`	Unreliable evidence or estimated data
`1`	Questionable reliability of evidence (interviews, census, oral genealogies, or potential for bias, such as an autobiography)
`2`	Secondary evidence, data officially recorded sometime after the event
`3`	Direct and primary evidence used, or by dominance of the evidence

Although the values look like integers, they do not have numeric meaning.

> *Note* — The structures for representing the strength of and confidence in various claims are known to be inadequate and are likely to change in a future version of GEDCOM.

`RESN`

Value	Meaning
`CONFIDENTIAL`	This data was marked as confidential by the user.
`LOCKED`	Some systems may ignore changes to this data.
`PRIVACY`	This data is not to be shared outside of a trusted circle, generally because it contains information about living individuals.

When a List (p.23) of `RESN` (p.72) enumeration values are present, all apply.

> *Example* — The line `1 RESN CONFIDENTIAL, LOCKED` means the superstructure's data is both considered confidential *and* read-only.

`ROLE`

Value	Meaning
`CHIL`	Child
`CLERGY`	Religious official in event; implies `OFFICIATOR`
`FATH`	Father; implies `PARENT`
`FRIEND`	Friend
`GODP`	Godparent or related role in other religions
`HUSB`	Husband; implies `SPOU`
`MOTH`	Mother; implies `PARENT`
`MULTIPLE`	A sibling from the same pregnancy (twin, triplet, quadruplet, etc). A `PHRASE` can be used to specify the kind of multiple birth.
`NGHBR`	Neighbor
`OFFICIATOR`	Officiator of the event
`PARENT`	Parent
`SPOU`	Spouse

Value	Meaning
`WIFE`	Wife; implies `SPOU`
`WITN`	Witness
`OTHER`	A role not listed here; should have a `PHRASE` substructure

These should be interpreted in the context of the recorded event and its primary participants. 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` (p.66)."

`SEX`

Value	Meaning
`M`	Male
`F`	Female
`X`	Does not fit the typical definition of only Male or only Female
`U`	Cannot be determined from available sources

This can describe an individual's reproductive or sexual anatomy at birth. Related concepts of gender identity or sexual preference are not currently given their own tag. Cultural or personal gender preference may be indicated using the `FACT` (p.64) tag.

`FAMC`.`STAT`

Value	Meaning
`CHALLENGED`	Linking this child to this family is suspect, but the linkage has been neither proven nor disproven.
`DISPROVEN`	There has been a claim by some that this child belongs to this family, but the linkage has been disproven.
`PROVEN`	There has been a claim by some that this child does not belong to this family, but the linkage has been proven.

> *Note* — The structures for representing the strength of and confidence in various claims are known to be inadequate and are likely to change in a future version of GEDCOM.

(Latter-Day Saint Ordinance).`STAT`

These values were formerly used by The Church of Jesus Christ of Latter-day Saints for coordinating between temples and members. They are no longer used in that way, meaning their interpretation in GEDCOM documents is subject to individual user interpretation

Value	Meaning
`BIC`	Born in the covenant, receiving blessing of child to parent sealing.
`CANCELED`	Canceled and considered invalid.
`CHILD`	Died before eight years old.
`COMPLETED`	Completed, but the date is not known.
`EXCLUDED`	Patron excluded this ordinance from being cleared in this submission.
`DNS`	This ordinance is not authorized.
`DNS_CAN`	This ordinance is not authorized, and the previous ordinance is cancelled.
`INFANT`	Died before less than one year old, baptism or endowment not required.
`PRE_1970`	Ordinance was likely completed because another ordinance for this person was converted from temple records of work completed before 1970.
`STILLBORN`	Stillborn, so ordinances not required.
`SUBMITTED`	Ordinance was previously submitted.
`UNCLEARED`	Data for clearing the ordinance request was insufficient.

`NAME`.`TYPE`

Value	Meaning
`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.

Value	Meaning
`MARRIED`	Name was person’s previous married name.
`PROFESSIONAL`	Name used professionally (pen, screen, stage name).
`OTHER`	A name type not listed here; should have a `PHRASE` substructure

4. The GEDZIP file type

It is often useful to transmit a GEDCOM document together with a set of external files. The GEDZIP file format is provided for this purpose.

A GEDZIP file is a zip archive, as defined by http://www.pkware.com/appnote and standardized by ISO/IEC 21320-1:2015.

> *Note* — A few details about the zip archive format are useful to fully understand GEDZIP:
>
> - An archive can contain one or more files.
> - Files within an archive can be added, removed, or updated individually without needing to re-process the rest of the archive. Libraries such as libzip allow applications to operate directly on the zip archive as if it were a normal directory tree.
> - What the zip specification calls a "file name" is actually a local path and may contain directories.
> - Directory separators are `/` internally and are converted to the appropriate form by the zip processing tool during zipping and unzipping. Because of this, unzipping a GEDZIP in any local directory results in all GEDZIP `FILE` (p.64) references working as-is for the resulting `gedcom.ged` without the need for any additional processing.

Each GEDZIP file contains the following entries:

- An entry with name `gedcom.ged` containing a GEDCOM document.
- An entry for each *local file* `FILE` structure in `gedcom.ged`, with the same zip *file name* as the corresponding `FILE` payload.

All file names inside a GEDZIP are case-sensitive.

Many other zip-based file formats (such as jar, epub, docx, GEDCOM-X) assign special meaning to the zip directory `META-INF` and the zip file names `MANIFEST.MF` and `META-INF/MANIFEST.MF`. These have no special meaning in GEDZIP and it is recommended that they not be used in a GEDZIP file, both to avoid confusing systems that look inside zip archives to determine their file type, and to leave open the possibility of their addition in a future version of GEDCOM.

5. Contributors

This document was based on the GEDCOM 5.5.1 document, and could not have existed without the contributors to that and previous GEDCOM standards. Appreciation is extended to all family history participants that have made GEDCOM the *de facto* standard for saving and transferring genealogical information.

New contributions in this edition benefited from the input of a large number of people:

Managing Editors

- Gordon Clarke, **FamilySearch**
- Luther Tychonievich, **FHISO** and **University of Virginia**

GEDCOM Taskforce

- Gordon Clarke, **FamilySearch**
- David Pugmire, **FamilySearch**
- Jimmy Zimmerman, **FamilySearch**
- Larry Telford, **FamilySearch**
- Matt Misbach, **FamilySearch**
- Russell Lynch, **FamilySearch**
- Robert Raymond, **FamilySearch**
- Gaylon Findlay, **Ancestral Quest**
- Derek Maude, **Ancestry**
- James Tanner, **Arizona Genealogy**
- John Cardinal, **Family History Hosting**
- Luther Tychonievich, **FHISO** and **University of Virginia**
- Albert Emmerich, **GEDCOM-L**
- Dave Berdan, **Legacy Family Tree**
- Evgen Zherebniy, **Mackiev Software**
- Jason Fletcher, **Midlera Software**
- Uri Gonen, **MyHeritage**
- Dallan Quass, **OurRoots.org**
- Tony Proctor, **Proctor.net**
- Bill Harten, **Puzzilla**
- Bruce Buzbee, **RootsMagic**

GEDCOM Development Teams

- **Tags team**: Luther Tychonievich, Albert Emmerich, Russell Lynch, Tony Proctor, John Cardinal
- **Extensions team**: Luther Tychonievich, Tony Proctor, Jimmy Zimmerman
- **Notes team**: Dallan Quass, David Pugmire, Jason Fletcher, Russell Lynch
- **External Media team**: Dallan Quass, Jason Fletcher, Derek Maude
- **Hypothesis team**: Russell Lynch, Derek Maude, Michael Ritchey, Luther Tychonievich
- **Names and Places team**: James Tanner, Tony Proctor

6. Appendix A: Known Calendars and Dates

6.1. Known Calendars

GEDCOM 7 defines four calendars: `GREGORIAN`, `JULIAN` (p.89), `FRENCH_R` (p.89), and `HEBREW` (p.90). Previous versions of GEDCOM also provided for, but did not define the meaning of, `ROMAN` and `UNKNOWN` calendars.

Extension calendars should use the usual rules for extensions, including using `_` as the leading character of the calendar name. Month codes in extension calendars must either be already used for the same month name in another calendar or must start with `_`.

Each calendar must list its permitted epochs and their meaning.

`GREGORIAN`

The Gregorian calendar is the now-ubiquitous calendar introduced by Pope Gregory XIII in 1582 to correct the Julian calendar which was slowly drifting relative to the seasons.

Permitted months are

Code	Name
`JAN`	January
`FEB`	February
`MAR`	March
`APR`	April
`MAY`	May
`JUN`	June
`JUL`	July
`AUG`	August
`SEP`	September
`OCT`	October

Code	Name
`NOV`	November
`DEC`	December

The epoch marker `BCE` is permitted in this calendar; year *y* BCE indicates a year *y* years before year 1. Thus, there is no year 0; year 1 BCE was followed by year 1.

`JULIAN`

The Julian calendar was introduced by Julius Caesar in 45 BC and subsequently amended by Augustus in about 8 BC to correct an error in the application of its leap year rule during its first three decades. Years had been counted from various starting epochs during the Julian calendar's use; the version standardized by GEDCOM uses the same starting epoch as the Gregorian calendar.

This calendar uses the same months as the Gregorian calendar, differing only in which years February has 29 days.

The epoch marker `BCE` is permitted in this calendar; year *y* BCE indicates a year *y* years before year 1. Thus, there is no year 0; year 1 BCE was followed by year 1.

`FRENCH_R`

The French Republican calendar or French Revolutionary calendar are the names given to the new calendar adopted in 1794 by the French National Convention. This calendar was adopted on Gregorian day 22 September 1792, which was 1 Vendémiaire 1 in this calendar. It was abandoned 18 years later.

Permitted months are

Code	Name
`VEND`	Vendemiaire
`BRUM`	Brumaire
`FRIM`	Frimaire
`NIVO`	Nivose
`PLUV`	Pluviose
`VENT`	Ventose
`GERM`	Germinal

Code	Name
`FLOR`	Floreal
`PRAI`	Prairial
`MESS`	Messidor
`THER`	Thermidor
`FRUC`	Fructidor
`COMP`	Jour Complementairs

No epoch marker is permitted in this calendar.

`HEBREW`

The Hebrew calendar is the name given to the calendar used by Jewish peoples around the world which developed into its current form in the early ninth century. It traditionally marks new days at sunset, not midnight. Its first day (1 Tishrei 1) primarily overlapped with Gregorian 7 September 3761 BCE and Julian 7 October 3761 BCE (starting at sunset on the 6th day of those months).

Code	Name
`TSH`	Tishrei (תִּשְׁרֵי)
`CSH`	Marcheshvan (מַרְחֶשְׁוָן) or Cheshvan (חֶשְׁוָן)
`KSL`	Kislev (כִּסְלֵו)
`TVT`	Tevet (טֵבֵת)
`SHV`	Shevat (שְׁבָט)
`ADR`	Adar I, Adar Rishon, First Adar, or Adar Aleph (אדר א׳)
`ADS`	Adar (אֲדָר); or Adar II, Adar Sheni, Second Adar, or Adar Bet (אדר ב׳)
`NSN`	Nisan (נִיסָן)
`IYR`	Iyar (אִייָר)
`SVN`	Sivan (סִיוָן)
`TMZ`	Tammuz (תַּמּוּז)
`AAV`	Av (אָב)
`ELL`	Elul (אֱלוּל)

To keep the lunar-based months synchronized with the solar-based years, some years have Adar I and others do not, instead proceeding from Shevat directly to Adar II. However, in common (non-leap) years, it is common to simply write "Adar" not "Adar II", which users not aware of the distinction might incorrectly encode as `ADR` instead of `ADS`. It is recommended that systems knowing which years had Adar I and which did not replace `ADR` in common years with `ADS`.

No epoch marker is permitted in this calendar.

6.2. Dual dates

The day on which a new year began and the year number increased varied at different times and places during the use of the Gregorian and Julian calendars. For example, England measured the new year as 25 March until 1752, which it switched to 1 January. In periods of transition, or when writing after a change about dates occurring before a change, it was sometimes common to indicate 2 years with a slash, for example, "30 January 1648/49" meaning "1648 if you count the new year as coming after 30 January, 1649 if you count it as coming before 30 January". Other notations, such as abbreviations for phrases like "new style" and "old style", were also sometimes employed.

```
2 DATE 30 JAN 1649
3 PHRASE 30 January 1648/49
```

Many nations transitioned from using the Julian calendar to using the Gregorian calendar. This transition caused a change in dates by several days, which (depending on the date in question) could change the month and year as well. In periods of transition, or when writing after a change about dates occurring before a change, it was sometimes common to indicate 2 dates with slashes, for example "23/6 November/December 1907" meaning "Julian 23 November 1907, Gregorian 6 December 1907". Other notations, such as abbreviations for phrases like "new style" and "old style", were also sometimes employed.

```
2 DATE 6 DEC 1907
3 PHRASE 23/6 November/December 1907
```

Some documents also used slashes to indicate approximate dates, such as writing a birth year as "1903/4" when it was computed from a year-granularity age at a given date.

```
2 DATE BET 1903 AND 1904
3 PHRASE 1903/4
```

GEDCOM 5.3 through 5.5.1 had special syntax for recording the first of these 3 concepts with a slash in the year. However, because slashes appear in historical documents with all 3 of the above meanings, some users misused this notation to record the other 2 situations as well. The result is ambiguity in the intended meaning of the resulting data.

GEDCOM 7.0 removed the year slash notation; a `PHRASE` (p.70) substructure should be used instead to clarify meaning.

6.3. Calendars in date ranges and date periods

Calendars apply to the subsequent `date` (p.20) production, not to the entire `DateValue`. Hence,

- `DATE FROM 1670 TO 1800` means `DATE FROM GREGORIAN 1670 TO GREGORIAN 1800`
- `DATE FROM 1670 TO JULIAN 1800` means `DATE FROM GREGORIAN 1670 TO JULIAN 1800`
- `DATE FROM JULIAN 1670 TO 1800` means `DATE FROM JULIAN 1670 TO GREGORIAN 1800`

Because some systems may show dates as-is to users and not all users understand the above rule, it is *recommended* that `calendar` tags be included if any `date` is non-`GREGORIAN` (p.88). It is *recommended* that the `calendar` tag be omitted if all `date`s in a payload are in the Gregorian calendar. Hence, the recommended forms of the previous three dates are

- `DATE FROM 1670 TO 1800`
- `DATE FROM GREGORIAN 1670 TO JULIAN 1800`
- `DATE FROM JULIAN 1670 TO GREGORIAN 1800`