GenXML 3.0

16. June 2010

Contents

1 Why GenXML?

1.1 What is GenXML?

GenXML is a file format for exchanging data between genealogy programs. It is based on XML and defined by a XML schema. It is not intended to be used as an internal format of any genealogy programs, although it may be possible.

The idea of GenXML is that:

- It shall be easy to read by most genealogy programs.
- It shall be easy to write by most genealogy programs.
- It shall be easy to manipulate by third party programs.
- All kinds of information shall fit into one and only one place.

GenXML acknowledges the fact that there are both simple and advanced genealogy programs and that it may be used in the exchange of data both between two simple programs, between two advanced programs, from a simple program to an advanced program, and from an advanced program to a simple program.

The data exchange should be made without data loss, except in the case data is transferred from an advanced program to a simple program. All data that is impossible to import, should be written to a log file during the import. Se also chapter 5.

A single scan is always enough to interpret a GenXML file as opposed to Gedcom. GenXML includes no forward references.

GenXML is mainly inspired by the theoretical Gentech Genealogical Data Model (see www.gentech.org) and Gedcom Future Directions, which is an unfinished replacement of Gedcom 5.5. GenXML is somewhat simpler than the Gentech model.

1.2 Differences Between GenXML and Gedcom

There already exists a file format for data exchange between genealogy programs. That is Gedcom, defined by the LDS church. The latest version is 5.5 and is dated 2. January 1996.

There are several problems with Gedcom 5.5:

- It is not clearly defined and is often difficult to interpret.
- There are about as many variants of Gedcom as there are programs that use it.
- It is often unclear where to put data. Almost "everything" is legal.
- Gedcom does not build upon an evidence/conclusion model.
- There is no support for data connected to the research process.
- The main purpose of Gedcom is for sending data to LDS' Ancestral File database.
- Gedcom will only have minor updates.

Compared to Gedcom, GenXML is enhanced in several ways:

- It is easier to see what version of GenXML that is used.
- The division into levels make it more easy to understand the capabilities of a program and also helps "pushing" the program developers into upgrading their program.
- GenXML is based on an evidence/conclusion model.
- There is no limit of possible kinds of events or attributes.

- GenXML includes advanced name, place and address structures.
- The main purpose of GenXML is for exchanging data between amateur genealogy programs.

1.3 GenXML compared to the Gentech Genealogical Data Model (GDM)

Note that the objective of GenXML is completely different from that of GDM. GDM is a data model only, and not very detailed. GenXML is primarily a file format. One may look at the GenXML data model as a simplified version of GDM. You loose some constructions but get a model that is much easier to implement. Espescially the GROUP entity opens many possibilities, but makes it difficult to exchange data between applications that may use it in different ways. GenXML is also much more detailed than GDM. In addition, GenXML is, in a larger extent than GDM designed to support simpler data models also.

GenXML and GDM share many properties. For example both supports a source hierarchy and neither have a family entity, like Gedcom. But GenXML has a simpler research model than GDM.

GDM has an EVENT-TYPE, but no event class. The event class construction of GenXML makes it easier for applications to understand user-defined event types from other applications.

GDM's extensive use of PERSONAs may seem quite different from GenXML. It is not. GDM's ASSERTION entity corresponds to GenXML's *assertion* structure. While GDM's four subject types PERSONA, EVENT, GROUP and CHARACTERISTIC compares closely to *alias*, *event*, *relationship* and *attribute*, although the GROUP entity is much more flexible than the relationship structure. But in GenXML *alias* is regarded as an assertion on its own, and not just a property of assertions of other types. This is more convenient when entering much information on the same person from for example a family history.

1.4 Differences Between GenXML 1.0 and GenXML 2.0

GenXML 2.0 is designed to be simpler, easier to implement, and more flexible than GenXML 1.0. Its updated data model is a purer evidence/conclusion model.

The major differences between GenXML 1.0 and 2.0 are:

- The evidence data model has been enhanced and is now closer to the GDM evidence model while retaining compatibility with the simpler, but much more widely used, Gedcom model. As a result of this the *document* structure has been removed, and the *sourceref* and *excerpt* structures have been merged. A source hierarchy is also supported.
- The *data* substructure of the *person* structure has been replaced by an *assertion* structure which is now defined as a main structure. Some of the *assertion* substructures may be linked to more than one person.
- The *association* structure is renamed *relationship*.
- There are no longer an attribute class "residence". Instead there is now an event class "residence". The residence of a person should be considered as an event rather than an attribute, because 1) residence has no property as other attributes has, and 2) several people may share the same residence for the same period of time, as opposed to attributes that relates to only one person.
- The *children* structure (formerly substructure of *person*) is removed. The order of children is now taken care of by the *relationship* structure in the same way as other assertions.
- Both the *group* and *couple* structures have been removed. Instead events may relate to more than one principal person. A group or couple is from now on no more than two or more people participating in one or more common events.

1.5 Differences Between GenXML 2.0 and GenXML 3.0

GenXML 3.0 does not modify the main structure of GenXML 2.0. Some new features have been added, and some issues have been fixed.

Changes to the source model:

- GenXML now supports tabular source data. The columns are specified in the *source* structure while all source data are specified in *excerpt* structures, one record/row per *excerpt*.
- A *source* can now include lists of surnames and places relevant for that source. The lists are mainly useful for searching purposes.
- The 'text' tag has been removed from the *source* structure, as this tag allowed source data to be stored both in the *source* structure and in the *excerpt* structure. Now all source data can be stored in the *excerpt* structure only.
- A new attribute, *level*, has been added to the excerpt structure. This allows for keeping the history of imported data.

Changes to assertion data:

- A *relationship* can now contain only a single parent-child relationship. To record a father-child and a mother-child relationship, two *relationship* structures must be created.
- New event type classes: Engagement.
- A sortdate tag has been added to the *date* structure for sorting and calculation purposes.
- The era tag of the *simpledate* structure is now valid for the proleptic Gregorian calendar as well as the Julian calendar. This was not really a problem in GenXML 2.0, but according to the definition it was not valid for the Gregorian calendar.
- An optional 'maid' tag has been added to the *personalname* structure. This may be used for storing the maiden name in the case you want to keep the maiden name together with the name while it is no longer a part of that name.
- The *place* structure is now on top level and hierarchical. This allows for structured places.
- The tag 'alias' has been added to the *place* structure for connecting different place names representing the same physical place.
- Additional value for the tp attribute of the *place* structure: island.
- Additional value for the tp attribute of the np tag in the *name* structure: ordi.

Other changes

- The *header* structure has been enhanced with three additional tags: name, create, and backup.
- The *eventtype* structure has been enhanced with three additional tags: principalfmt, withnessfmt, and print. A new attribute, 'active', has also been added.
- Extensions are now allowed in all main structures as well as *object*, *address* and *place*.
- The 'solution' tag is now optional for the *objective* structure as it is for the *task* structure.
- New tags have been added to the *object* structure: *objtype*, *author*.
- The type of the *title* tag of the *object* structure has been changed from normalizedString to normstringlang.

2 Properties of GenXML

2.1 File extension

GenXML-files should preferably have the file extension '.gml'.

2.2 Versions

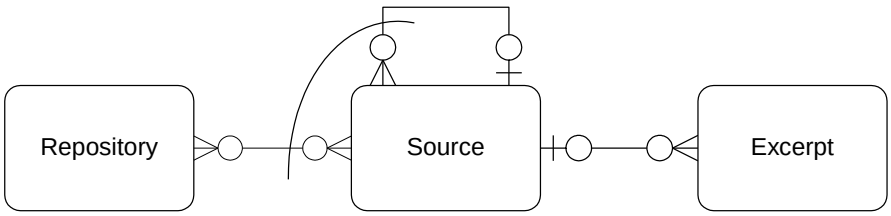
The first structure in the GenXML-file is the *file* structure which tells the parser which version of GenXML the file uses. It is not a goal to make later version compatible with the current one. That would probably be a very difficult task and also limit the quality of the new version. Later versions should therefore be regarded as separate formats, like Gedcom 4.0 and 5.5 also should be regarded as separate formats.

Note that the GenXML version also may be seen from the schema reference in the GenXML-file. This schema reference is not mandatory though.

2.3 Evidence and Conclusions

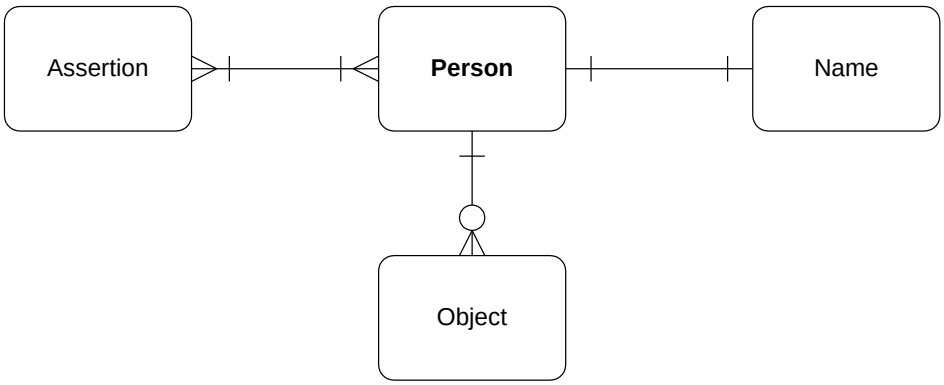
The GenXML format is based on an evidence/conclusion model.

2.3.1 The Evidence Submodel



The main structures of the evidence part is *repository*, *source* and *excerpt*. Evidence is what is found in existing (preferably primary) sources. The evidence is broken down into excerpts. The conclusions are based on these excerpts.

2.3.2 The Conclusion Submodel

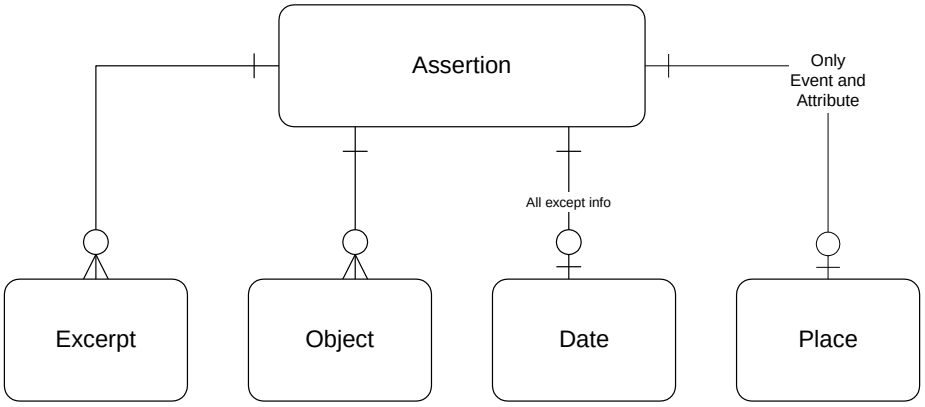


The conclusions are stored as *assertions*, and all assertions are linked together through the person structure which basically represents a single individual. There are several kinds of assertions:

- alias

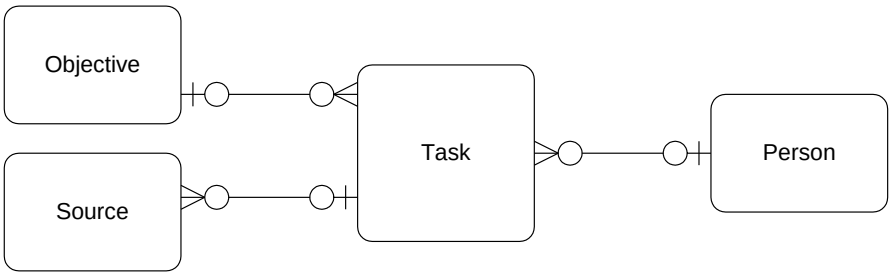
- relationship
- attribute
- event
- info

The different kind of assertions shares many properties, although there are some differences also:



2.4 The Research Process

The GenXML format also includes support for the research process. This includes the *objective* and *task* structures. The Gentech Genealogical Data Model also includes a project entity. This is not supported in GenXML, because the complete GenXML file is regarded as one project. This may not be satisfactory for expert systems that often have more than one user.



The *objective* structure holds one research objective. A research objective includes one or more research tasks. Each *task* may be related to one person.

Note that GenXML supports "independent" tasks – that is tasks not related to an objective. That is why *task* is a main structure and not a substructure of *objective*. This is for compatibility with programs that have only a simple todo structure. Preferably all tasks should be parts of a research objective.

2.5 The Note and Info Structures

The *note* structure is mainly used for short comments not meant for printing in reports. Notes meant to be printed in reports must be placed in the *info* structure (see definition in 4.5.5).

3 Technical Details

3.1 Character Set

All character sets that may be used for XML-files in general, may also be used for GenXML. That includes Unicode. However, it is not likely that most systems will support all character sets, and many systems will not support Unicode at all. It is therefore recommended that GenXML-files use ISO-8859-1 as this closely matches the ANSI character set used by Windows. Applications should at least support ISO-8859-1, but other character sets, like UTF-8 and UTF-16, should also be supported. Preferably the user may choose character set when exporting GenXML.

Note that the characters '&' and '<' are reserved and must never appear in character data. They must be replaced by '&' and '<'. (See section 2.4 in Extensible Markup Language (XML) 1.0 (Second Edition).)

3.2 Definition of Datatypes

All simple datatypes, except 'ident', are used as defined in the W3C XML Schema recommendation (see http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/).

Datatype	Definition
any	Any tag or structure (i.e. SimpleType or ComplexType in the XML Schema terminology) from other namespaces. Consequently, tags or structures defined in the GenXML Schema are not allowed here. The 'any' type is used to open up for extensions.
anyURI	A uniform resource identifier reference (URI).
Base64Binary	Represents Base64-encoded arbitrary binary data as defined in section 6.8 in http://www.w3.org/TR/xmlschema-2/#RFC2045
Boolean	Two possible values: "true" or "false".
Datetime	A date basically on the form YYYY-MM-DD**T**tt:mm:ss. Parts of it may be omitted so that the following forms may be used (see the XML Schema specifications for complete definition): YYYY YYYY-MM YYYY-MM-DD YYYY-MM-DD**T**tt:mm YYYY-MM-DD**T**tt:mm:ss
ident	This type is used as a unique identifier for each main structure. It consists of a prefix followed by positive number less than 2.000.000.000. The prefixes are as follows: "R" – repository "S" – source "X" – excerpt "E" – eventtype "P" – person "L" – place "A" – assertion "O" – objective "T" – task
int	A 32-bit signed integer.
Language	Identifies the language of the tag in which it is defined. Legal values are language codes as defined by ISO 639, a combination of such language codes and country codes (ISO 3166-1) as "en-US", or IANA-LANGCODES (see http://www.isi.edu/in-notes/iana/assignments/languages/).
normalizedString	A string without any carriage return (0x0D), line feed (0x0A) or tab (0x09) characters. Space characters must not be repeated.

String	A string of unlimited length.
Token	A string without any carriage return (0x0D), line feed (0x0A) or tab (0x09) characters. There must be no leading or trailing spaces and space characters must not be repeated.

4 GenXML Specifications

The following operators are used:

- ?: The field/structure must be included zero or one time.
- *: The field/structure may be included any number of times.
- +: The field/structure must be included one or more times.

All XML files starts with a header that may include information on the XML version, the character set used, the document type definition or schema used, stylesheets for visual formatting and more. This may for example look like this:

<?xml version="1.0" encoding="ISO-8859-1"?>

Definition of the main structure of a GenXML file:

<genxml>
content: (file, header, repository*, source*, excerpt*, eventtype*, person*, place*, assertion*, objective*, task*, total)
</genxml>

Note that genxml, as the main structure, optionally may include standard XML attributes defining the schema (see the examples below).

Content of genxml:

Element	Type	Level
file	file structure, see section 4.1.1	1
header	header structure, see section 4.1.2	1
repository	repository structure, see section 4.1.3	3
source	source structure, see section 4.1.4	2
excerpt	excerpt structure, see section 4.1.5	2
eventtype	eventtype structure, see section 4.1.6	1
person	person structure, see section 4.1.7	1
place	Place structure, see section 4.1.8	1
assertion	assertion structure, see section 4.1.9	1
objective	objective structure, see section 4.1.10	4
task	task structure, see 4.1.11	3
total	total structure, see section 4.1.12	1

Examples:

```
<genxml>
   …
</genxml>

<genxml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
➔xsi:nonamespaceSchemaLocation="genxml21.xsd">
   …
<genxml>
```

4.1 Main Structures

4.1.1 File

The file structure is the very first structure of the GenXML-file. This tells the parser which version of the GenXML format the file uses and at what level it is. Different versions must be regarded as separate formats. If an application only understands the GenXML 2.0 format, it should not import a version 2.1 file – at least not without warning the user.

The level indicates what kind of data the importing application may expect. The file should not include data of higher levels than the given level. For example a level 2 compliant application (see section 1) using repositories, but no other data of level 3 or higher, will export a level 3 file. A level 2 file must not include any repositories.

Definition:

<file>
content: (version, level)
</file>

Content of file:

Element	Type/description	Level
version	int: The GenXML-version * 100. For example is version 1.0 coded as 100 and 1.1 as 110. For GenXML version 2.1 the value of this field must always be 210.	1
level	int: The highest level of which data are included. If, for example, *objective* structures are included in the file, then the file is a level 4 file, even if the exporting application is only level 2 compliant. This tells the importing application what it might expect to find in the file.	1

Example:

```
<file>
   <version>200</version>
   <level>2</level>
</file>
```

4.1.2 Header

The header structure includes information about the system that created the GenXML file, and about the owner of the database.

Definition:

<header>
content: (exportingsystem, version, exported, name?, language?, owner?, copyright?, note?, create?, change?, backup?)
</header>

Content of header:

Element	Type/description	Level
exportingsystem	normalizedString: Name of the exporting application.	1
version	normalizedString: Version of the exporting application.	1
exported	datetime: The date of the export.	1
name	string: Name of the database	2
language	language: The main language of data in this file.	2
owner	owner structure (see section 4.3.1): Information on the owner of the exported database.	N/A
copyright	normalizedString: A copyright message of the exported database.	N/A
note	string: A description of the database.	N/A
create	datetime: The date and time of the creation of the database. This is not the same as the creation of the GenXML file.	N/A
change	datetime: The date and time of the last change of this database.	N/A
backup	Datetime: The date and time of the last backup.	N/A

Example:

```
<header>
   <exportingsystem>Slekten</exportingsystem>
   <version>0.8</version>
   <exported>2001-11-25</exported>
   <language>en</language>
   <owner>
      …
   </owner>
   <copyright>2001 Christoffer Owe</copyright>
   <note>This is a test file.</note>
</header>
```

4.1.3 Repository

The repository structure represents a repository of sources, like a library or a public record office.

Definition:

<repository>
attributes: (id, lang?)
content: (name, address?, email?, uri?, note?, change?, ext*)
</repository>

Attributes of repository:

Element	Type/description	Level
id	ident: Unique identifier	3
lang	language: The language of the repository information.	3

Content of repository:

Element	Type/description	Level
name	normalizedString: The name of the repository.	3
address	address structure (see section 4.2.4): The address of the repository.	3
email	normalizedString: Email address of the repository.	3
uri	anyURI: World wide web homepage of the repository.	3
note	stringlang structure (see section 4.2.1): Note.	3
change	datetime: The date and time of the last change of this repository information.	N/A
[ext]	any: Any simple or complex type of other namespaces	N/A

Example:

```
<repository id="R5" lang="no">
   <name>Deichmanske bibliotek</name>
   <address>
      <ap>Henrik Ibsensgt. 1</ap><lf/><lf/>
      <ap>N-0179 Oslo</ap><lf/>
      <ap>Norway</ap>
      <phone>23 43 29 00</phone>
      <fax>22 11 33 89</fax>
   </address>
   <email>deichman@deich.folkebibl.no</email>
   <uri>http://www.deich.folkebibl.no</uri>
</repository>
```

4.1.4 Source

The source structure represents a book, a collection of documents, or similar. It may be printed, hand-written, in electronic form, or on microfilm/-fiche.

Definition:

<source>

attributes: (id, class?, kind?, media?, lang?)
content: (author, title, shorttitle, published?, (isbn | issn)?, (repositoryref* | sourceref)?, template?, sourname*, place*, object*, note?, change?, ext*)
</source>

Attributes of source:

Element	Type/description	Level
id	ident: Unique identifier	2
class	"series": This is a series consisting of more than one volumes. It may also be a periodical.	4
	"book" (default): This is a single book or a single volume of a series.	4
	"issue": This is an issue of a periodical, for example a magazine.	4
	"article": This is an article, in for example a magazine.	4
	"part": This is a part of a book, consisting of more than one chapter or documents.	4
	"chapter": This is a chapter.	4
	"document": This is a document, such as a birth certificate, a census entry, a church register entry, or an inscription like a tombstone.	4
	"section": This is a part of a chapter.	4
	"excerpt": This is a single excerpt, consisting of one or a few sentences.	4
kind	"original": This is an original source.	4
	"reprint": This is a reprint of the original source.	4
	"copy": This source is a copy or transcript of the original source.	4
	"unknown" (default): It is not known if this source is an original or a copy.	2
media	"audio": This source exist as sound only.	4
	"digital": This source exists as a digital file.	4
	"fiche": This source is published on microfiche.	4
	"handwritten": This source is handwritten.	4
	"inscription": This source is an inscription, for example on a tombstone.	4
	"microfilm": This source is published on microfilm.	4
	"photo": This source exists as a photography.	4
	"printed": This is a printed source.	4
	"video": This source exist as video.	4
	"other": This source exists on some other kind of media than the ones described above.	4
	"unknown" (default): The media of this source is unknown.	2
lang	language: The language of the source.	3

Content of source:

Element	Type/description	Level
author	normalizedString: The name of the source's author or editor.	2
title	normalizedString: The complete title of the source.	2
shorttitle	normalizedString: A short version of the title.	2
published	normalizedString: Publication facts such as where the source was published, the name of the publisher, and the year it was published.	2
isbn	normalizedString: International Standard Book Number. A 10-digit number which identifies any published book or edition.	4
issn	normalizedString: International Standard Serial Number. A 8-digit number which identifies any periodicals.	4
repositoryref	repositoryref substructure (see section 4.6.1): Includes a pointer to a repository structure.	3
sourceref	ident: Pointer to a 'higher' source record representing a source of	4

	which this source is a part. For example this source may represent an specific volume, while the 'higher' source represents the entire series.	
template	string: Specifies a template used for the excerpt text for all excerpts for this source. The template should be on the form "col1;col2;col3". The excerpt text should have the same form. If the source contains a table of data, this field represents the headings of the table columns while the each excerpt represent one table row.	4
surname	string: A surname covered or mentioned by this source.	4
place	ref: Points to a place structure (see section 4.1.8) representing a place covered or mentioned by this source.	4
object	object structure (see section 4.2.3): An object related to the document. (For example a picture of the document.)	3
note	stringlang structure (see section 4.2.1): Note.	2
change	datetime: The date and time of the last change of this source information.	N/A
[ext]	any: Any simple or complex type of other namespaces	N/A

Some sources contain tabular data. For example, in church records you may find the following table for baptisms:

Date	**Name**	**Parents**	**Address**
12/3 1887	Ole	Sven Olsen and Marie Hansdatter	Sandnes
14/3 1887	Ingrid	Peder Larsen and Ingeborg Rasmusdatter	Sandnes

To store this table in GenXML 2.1, the source template feature must be used. In the source structure, include the following:

```
<template>Date;Name;Parents;Address</template>
```

Two excerpts are needed – one per row in the table. The excerpt for the first row should be coded as follows:

```
<excerpt id="X3">
      <text>12/3 1887;Ole;Sven Olsen and Marie Hansdatter;Sandnes</text>
      <sourceref>S1</sourceref>
</excerpt>
```

If any part of the text include semicolon, that part of the text can be enclosed by quotes, following the standard rules for csv-data.

If one single source includes several different tables – which is often the case – the source hierarchy feature should be used as well as the template feature. Each table should be represented by a source structure. In addition a source record representing the entire source should be created. The other sources should be linked to this 'overall' source by the sourceref-tag.

Note that in version 2.0, source text could be included in a 'text' field. In version 2.1, all source text should be contained in excerpts. If converting from GenXML 2.0 to 2.1, an additional excerpt should be generated for source text contained directly in the source structure.

4.1.5 Excerpt

The excerpt structure represents a single excerpt (like "John was the third son of Robert" or "John was 64 years old in 1759") extracted from a source.

Definition:

<excerpt>
attributes: (id, lang?)
content: (text?, quality?, page?, sourceref, level?, note?, change?, ext*)
</excerpt>

Attributes of excerpt:

Element	Type/description	Level
id	ident: Unique identifier	2
lang	language: The language of the excerpt.	3

Content of excerpt:

Element	Type/description	Level
text	string: Text extracted from the referred document. This field is expected to hold csv-data (one single row) if the template tag has been included in the corresponding source structure. For more information see section 4.1.4.	3/4
quality	an integer between 0 and 100: This is the quality of the document or data from the referred source, in percents of reliability. If this is not present, the quality must be regarded as unknown..	3
page	normalizedString: A short description of where in the source the referred data exist.	2
sourceref	ident: Pointer to a source structure.	2
level	int: Indicates if the source has been verified directly or indirectly. If the excerpt has been entered directly, the value should always be 0. If the excerpt has been imported, it's level should be increased by 1. See below for more information. Default value is 0.	4
note	stringlang structure (see section 4.2.1): Note.	3
change	datetime: The date and time of the last change of this excerpt information.	N/A
[ext]	any: Any simple or complex type of other namespaces	N/A

4.1.5.1 Quality

The following table shows the recommended conversion between "quality" and Gedcom QUAY:

Gedcom QUAY-value	GenXML "quality"
0 – unreliable evidence	0-49%
1 – questionable reliability of evidence	50-79%
2 – secondary evidence	80-94%
3 – primary evidence	95-100%

4.1.5.2 Level

When importing a GenXML or Gedcom file into a database, several sources and excerpts may be imported with it. These sources have not been verified directly. Instead, for the user, the imported file itself is the source, with several unverified excerpts. The importing application should therefore record the file itself as a source and add excerpts referencing this source, for all assertions imported from that file. These excerpts will be of level 0. All excerpts imported from the file should get their level increased by one (meaning the value should be one if no level is specified in the file).

The process may be repeated if the database, including the imported data, is exported and imported into another database. The original excerpts will then be of level 2, and the importing history is retained.

An application should reset the level if the excerpt is changed, assuming the user has verified it.

4.1.6 EventType

The *eventtype* represents a certain event type (such as marriage, birth, retirement), user defined or application defined.

Note that the class is always a noun while the description would typically be a verb (birth – born, baptism – baptized).

The description field makes event types language dependant. It may seem a drawback that the event types can't easily be translated from a language to another. But event types may be quite different in different cultures, and an automatic translation may not retain the correct meaning. It is therefore important to understand that the event types is a part of the database as well as other genealogical data, and the database can only be translated by a qualified person.

Definition:

<eventtype>
attributes: (id, class, active?, lang?)
content : (description, gedcomtag ?, roles ?, principalfmt ?, withnessfmt ?, print ?, note ?, ext*)
</eventtype>

Attributes of eventtype:

Element	Type/description	Level
id	ident: Unique eventtype identifier.	1
class	"adoption": Adoption or similar events (like moving in with foster parents).	3
	"annulment": The annulment of a marriage.	3
	"baptism": Any kind of baptism or similar events.	2
	"birth": The birth of the person.	1
	"blessing": Any kind of blessing or similar events.	3
	"burial": Any kind of burial (for example funeral or interment).	2
	"census": Any kind of events dealing with the counting of people.	3
	"confirmation": Any kind of confirmation or similar events regardless of religion.	3
	"coronation": Any kind of coronation or similar events.	3
	"cremation": The cremation of a person.	3
	"death": The death of a person.	1
	"discharge": The event of leaving the army.	3
	"divorce": The event of dissolving a marriage or similar union.	2
	"election": The event of being elected or similar.	3
	"emigration": The event of leaving the place where one live.	3
	"engagement": The event of uniting two persons with the intention of later marriage, or a similar kind of union that should not be treated as marriage.	3
	"enlistment": Any kind of enlistment to the army, or similar events.	3
	"graduation": The event of graduating from a school or university, or similar.	3

	"health": Any event related to health, for example vaccination or operations.	3
	"immigration": The event of entering a new locality with the intention of residing there.	3
	"internment": The event of being interned, or similar.	3
	"marriage": The event of uniting two persons in marriage or similar union. See also "engagement".	1
	"naturalization": The event of obtaining citizenship in a city or a country, or similar events.	3
	"ordination": The event of receiving authority to act in religious matters, or similar events.	3
	"residence": The address or place where one or more persons live(d).	3
	"retirement": Any kind of retirement.	3
	"other": Any other kind of events.	3
active	"yes" (default) or "no": Specifies if new events od this type may be created.	4
lang	language: The language of the eventtype information.	4

Content of eventtype:

Element	Type/description	Level
description	normstringlang structure (see section 4.2.2): Short description of the event type.	3
gedcomtag	token (Default is "EVEN"): The GEDCOM 5.5-tag of this event type.	3
roles	int (Default is "1"): The maximum number of principal roles in this event type. If this value is set to "0", it means that events of this type may have an unlimited number of principal roles. Note that GenXML can't guarantee that no events of this type have more principal participants than specified here. But an application which doesn't follow this rule, can't expect its data to be imported correctly by other applications.	1/3
principalfmt	format: Format description for formatting sentences of events of this type for principal roles.	5
withnessfmt	format: Format description for formatting sentences of events of this type for withness roles.	5
print	Boolean: Flag defining if events of this type should be included in reports or not.	5
note	stringlang structure (see section 4.2.1): Note.	4
[ext]	any: Any simple or complex type of other namespaces	N/A

Example:

```
<eventtype id="E5" class="baptism" lang="en-us">
   <description lang="en-us">baptized</description>
   <gedcomtag>CHR</gedcomtag>
   <roles>1</roles>
   <note>A common christian baptism</note>
</eventtype>
```

4.1.7 Person

The *person* structure represents a single individual. If, however, there are two persons, each with their own data, that may be the same individual, there may be created a third *person* that, through the *subpersons* substructure, combines the two persons into one. If it is certain that the two person records really represents the same individual, they should be merged into one instead of using the *subpersons* structure. Note that no data except a name may be connected to a *person* record using the *subpersons* structure.

Definition:

<person>
attributes: (id, sex, lang?)
content: (personalname, (subpersons | (excerptref*, object*)), note?, change?, ext*)
</person>

Attributes of person:

Element	Type/description	Level
id	ident: Unique identifier	1
sex	"male": This person is a man.	1
	"female": This person is a woman.	1
	"unknown": The sex of this person is unknown.	3
lang	lang: The language of the person information.	3

Content of person:

Element	Type/description	Level
personalname	personalname structure (see section 4.2.5): Name of the person. This may be a normalized version of the persons name. The name as found in sources is stored in the alias record.	1
subpersons	subpersons structure (see section 4.4.1): A structure used for combining two persons into one. See section 6.2 for more information.	4
excerptref	ident: Pointer to an excerpt structure on which the information on this person is based.	2
object	object structure (see section 4.2.3): Files/objects related to this person.	3
note	stringlang structure (see section 4.2.1): Researcher notes for this person. This element should not contain genealogical data. Comments meant to be printed on reports (for example the life of the person) should be stored in info structures (see 4.5.5).	3
change	datetime: The date and time of the last change of this person information.	N/A
[ext]	any: Any simple or complex type of other namespaces	N/A

Example:

```
<person id="P57" sex="male">
   <personalname>
      <np tp="unkw">Fred</np>
      <np tp="surn">Lunde</np>
   </personalname>
   <change>2003-05-25</change>
</person>
```

4.1.8 Place

The place structure represents a place where some event happened. It is not intended for postal addresses.

Places are stored with name parts in separate records. What in Gedcom would be stored as a single place, like "Balquholly Castle, Aberdeenshire, Scotland", would in GenXML 2.1 be stored as three separate records, for "Scotland", "Aberdeenshire" and "Balquholly Castle". Note that the most significant part (here "Scotland") should always be stored first.

The place structure may and should be hierarchic, but doesn't have to. With flat data, each place may actually be stored as a separate hierarchy (as in the example above).

Definition:

<place id?>
attribute: (id?, tp?, lang?)
content: (prefix?, name, place*, alias?, date?, cords?, ext*)
</place>

Attribute of place:

Element	Type/description	Level
id	Ident: Unique identifier.	1
tp	"continent": The place name refers to a continent or part of the world.	4
	"country": The place name refers to a country.	4
	"state": The place name refers to a part of a country which are partly independent, like a state in USA or a greater principality.	4
	"county": The place name refers to a county or province, a part of a country or state, or a lesser principality, with its own authorities.	4
	"town": The place name refers to a town or city.	4
	"muni": The place name refers to a municipality.	4
	"citypart": The place name refers to a part of a town or city.	4
	"building": The place name refers to a building or a group of buildings, like a church or monastery.	4
	"diocese": The place name refers to an ecclesiastical district under the jurisdiction of a bishop.	4
	"deanery": The place name refers to a district under the jurisdiction of a dean, part of a diocese.	4
	"parish": The place name refers to an ecclesiastical district with its own church.	4
	"farm": The place name refers to a farm or estate.	4
	"ocean": The place name refers to an ocean or sea of salt water.	4
	"lake": The place name refers to a lake.	4
	"mountain": The place name refers to a mountain.	4
	"island": The place name refers to an island.	4
	"cemetery": The place name refers to a cemetery.	4
	"other": The place name refers to an other kind of geographical area or jurisdiction than the one above.	4
	"unknown" (default): The place name is of an unknown type or the exporting program does not support the above place name part descriptions.	1
lang	lang: The language in which the place name is written.	4

Content of place:

Element	Type/description	Level
prefix	normalizedString: Prefix used when printing place names, like "in", "at" or "near".	4
name	normalizedString: Place name. It can contain only a single place name (in GenXML 2.0 called place name part). Combined names, like "Skanderborg, Jylland, Denmark" must be stored as a hierarchy of several place structures. See example below.	1
place	ref: Places, being a part of this place	
alias	ref: Pointer to another instance of the same physical place, for example a different name valid in a different historic period or the same name elsewhere in the hierarchy (for example diocese and parish instead of county and town).	4
date	date structure (see section 4.2.7): The period when this place name was valid.	4
cords	coords structure (see 4.2.6): The geographical coordinates of the place.	4
[ext]	any: Any simple or complex type of other namespaces	N/A

In contradiction to all other top level structures, the id attribute is optional for places. It the place is to be referenced, it must have an id. As places are hierarchical, all leaves in the hierarchy will typically have an id, but not necessarily the nodes.

4.1.8.1 Examples

Example of level 1 place:

```
<place>
   <name>Scotland</name>
   <place>
      <name>Aberdeenshire</name>
      <place id="L14">
         <name>Balquholly Castle</name>
      </place>
   </place>
</place>
```

Example of level 4 place:

```
<place tp="state">
   <prefix>in</prefix>
   <name>Scotland</name>
   <place tp="county">
      <prefix>in</prefix>
      <name>Aberdeenshire</name>
      <place id="L14" tp="building">
         <prefix>near</prefix>
         <name>Balquholly Castle</name>
      </place>
   </place>
</place>
```

Example of place with alias and date:

```
<place tp="state">
   <prefix>in</prefix>
   <name>Scotland</name>
   <place tp="county">
      <prefix>in</prefix>
      <name>Aberdeenshire</name>
      <place id="L14" tp="building">
         <prefix>in</prefix>
         <name>Balquholly Castle</name>
         <date><to>1727</to></date>
      </place>
      <place id="L15" tp="building">
         <prefix>in</prefix>
         <name>Hatton Castle</name>
         <alias>L14</alias>
      </place>
   </place>
   <place tp="county">
      <prefix>in</prefix>
      <name>Banffshire</name>
      <place id="L16" tp="building">
         <prefix>in</prefix>
         <name>Hatton Castle</name>
         <alias>L14</alias>
         <date><from>1890</from><cto>1975</to></date>
      </place>
   </place>
</place>
```

4.1.8.2 More on Place Hierarchies

Some applications support place hierarchies. Others don't. GenXML may be used for both types of application as described in the following table:

		Importing appl.	
		Hierarchies supported	Hierarchies not supported
Exporting appl.	Hierarchies supported[1]	The places are both stored and imported as a true hierarchy.	The places are stored as a hierarchy, but each place name is imported and stored in a flat structure. As the importing application can't easily know if a place name is a node or a leaf (or both) in the hierarchy, all names should be regarded as place names (including all parent names to create place names of the Gedcom type). The alternative would be to read the complete hierarchy into memory to find the leaves and only store those.
	Hierarchies not supported	The places are stored in a flat structure, meaning separate hierarchies for each Gedcom type place name. The data may be automatically consolidated by the importing application based on place name and type, but it should be done very carefully not to mix different places with the same name.	The places are stored in a flat structure, meaning separate hierarchies for each Gedcom type place name. Each "hierarchy" must be reassembled into a single Gedcom type place name by the importing application. This safest way to do this would be to read all the hierarchies into memory to find all the lowest level nodes (leaves) and then store only those, or to store only place names actually referred to by the assertions, based on the capabilities of the application.

Place names doesn't have to be fully qualified. The following hierarchy is fully valid:

World
United Kingdom
Scotland
Aberdeenshire
Balquholly Castle
Caithness
Freswick
Balqhuolly Castle
Balquholly Castle

In the example above there are three place names "Balquholly Castle". One in Aberdeenshire and one in Caithness. The third is not specified. It may be the one in Aberdeenshire, the one in Caithness or a third one. The user doesn't always have enough information to decide, so partly and inaccurate information must be accepted.

[1] Please note that even if an application supports place hierarchies, it may choose (based on selections done by the user) to export places in a flat structure to improve the import into an application not supporting hierarchies.

A place may be a part of another place. It can also be in more than one place. A city may for example be in both a county and in a diocese. These assignments may also vary over time. This is not directly supported by GenXML. A city in a county and the same city in a diocese will be treated as two separate places in the place hierarchy. It is thus the place name reference in an assertion which decides which of these two instances of the same city to be used. The cities may still be identified as the same city by using coordinates.

4.1.9 Assertion

The *assertion* represents a piece of conclusional data for a person, such as the birth date, hair color or relationships.

Definition:

<assertion>
attributes: (id, datatype?, lang?)
content: ((alias | relationship | attribute | event | info),
(excerptref* | (assertionref, assertionref)), object*, note?, change?, ext*)
</assertion>

Attributes of assertion:

Element	Type/description	Level
id	ident: Unique assertion identifier.	1
datatype	"public" (default): Indicates "public" information that will be printed in reports and exported.	4
	"family": The data is public to family members only.	4
	"immfamily": The data is public to the immediate family only.	4
	"private": Indicates private or confidential information that will not normally be printed in reports or exported.	4
	"info": Indicates information that will normally not be printed in reports, not because it is private, but because it is mostly of interest to the researcher only.	4
lang	lang: The language of the assertion information.	3

Content of assertion:

Element	Type/description	Level
alias	alias structure (see section 4.5.1): The assertion is a name alias.	3
relationship	relationship structure (see section 4.5.2): The assertion represents a parent-child relationship.	1
attribute	attribute structure (see section 4.5.3): The assertion represents an attribute of the associated person(s).	1
event	event structure (see section 4.5.4): The assertion represents an event (like birth, death, marriage or divorce).	1
info	stringlang structure (see section 4.2.1): This is a string that may hold general information about a person that does not fit into any of the other assertion structures.	1
excerptref	ident: Pointer to an excerpt structure on which this assertion is based.	2
assertionref	ident: Pointer to an assertion on which this assertion is based. An assertion may be based on two other assertions instead of an excerpt.	4
object	object structure (see 4.2.3): Files/objects related to this assertion.	3
note	stringlang structure (see section 4.2.1): Researcher notes for this assertions. This element should not contain genealogical data.	3
change	datetime: The date and time of the last change of this couple information.	N/A

[ext]	any: Any simple or complex type of other namespaces	N/A

4.1.10 Objective

The objective structure represents a research objective. A research objective consists of one or more research tasks (see section 4.1.11). The research objective is split into one task for each affected person.

Note that the research objective may be seen as having a status, like the research task. The objective's status should then equal the lowest status of the objective's tasks. A application may easily calculate this. The research objective is not completed until all of its tasks are completed.

The task structure should be regarded as a substructure of the objective structure. But since the task structure is a level 3 structure and objective is a level 4 structure, the task structure is implemented as a main structure.

Definition:

<objective>
attributes: (id, lang?)
content: (title, problem, solution?, priority?, created?, change?, ext*)
</objective>

Attributes of objective:

Element	Type/description	Level
id	ident: Unique identifier	4
lang	lang: The language of the research objective information.	4

Content of objective:

Element	Type/description	Level
title	string: Short description of the research objective.	4
problem	string: Complete description of the research objective. Note that details regarding a single group, couple or person should be stored in the corresponding task structure.	4
solution	string: Complete description of the solution of the problem. Note that details regarding a single group, couple or person should be stored in a corresponding task structure.	4
priority	"low": The research objective has low priority.	4
	"medium": The research objective has medium priority.	4
	"high": The research objective has high priority.	4
created	datetime: The date of the creation of the research objective.	4
change	datetime: The date and time of the last change of this research objective.	N/A
[ext]	any: Any simple or complex type of other namespaces	N/A

4.1.11 Task

The task structure represents a research task. A research task may be related to a specific person, and to a specific source. It may (and should) also be part of a research objective.

Note that each combination of person and source should be separate tasks.

Definition:

<task>

attribute: (id, lang?)
content: (title, problem, solution?, objectiveref?, personref?, sourceref?, status, statusdate?, change?, ext*)
</task>

Attributes of task:

Element	Type/description	Level
id	ident: Unique identifier	3
lang	lang: The language of the research task information.	3

Content of task:

Element	Type/description	Level
title	string: Short description of the problem.	3
problem	string: Complete description of the problem and what need to be done.	3
solution	string: Description of the solution. This should be used for archiving purposes.	3
objectiveref	ref: Pointer to a objective structure. The task should always be a part of an objective if the objective structure is supported by the exporting program.	4
personref	ref: Pointer to a person structure.	3
sourceref	ref: Pointer to the source that is to be searched.	3
status	"new": This task is registered, but work is not started.	3
	"analysis": The work on this task has started.	3
	"finished": The work on this task is finished.	3
	"updated": The database has been updated with the results from the work on this task, and the case is closed.	3
	"rejected": The task is rejected and will not be executed.	3
statusdate	datetime: The date of the last status change. This value is system generated and not typed in by the user.	3
change	datetime: The date and time of the last change of this research task.	N/A
[ext]	any: Any simple or complex type of other namespaces	N/A

4.1.12 Total

The TOTAL structure is always the last one in the file and must always be included. The purpose is to tell the importing program how many structures that should have been imported.

Definition:

<total>
content: (repositories, sources, excerpts, eventtypes, persons, places, assertions, objectives, tasks)
</total>

Content of total:

Element	Type/description	Level
repositories	int: The total number of repository structures in the file.	1
sources	int: The total number of source structures in the file.	1
excerpts	int: The total number of excerpt structures in the file.	1
eventtypes	int: The total number of eventtype structures in the file.	1
persons	int: The total number of person structures in the file.	1
places	int: The total number of place structures in the file.	1
assertions	int: The total number of assertion structures in the file.	1
objectives	int: The total number of objective structures in the file.	1
tasks	int: The total number of task structures in the file.	1

Example:

```
<total>
   <repositories>1</repositories>
   <sources>4</sources>
   <excerpts>0</excerpts>
   <eventtypes>12</eventtypes>
   <persons>124</persons>
   <places>32</places>
   <assertions>49</assertions>
   <objectives>0</objectives>
   <tasks>0</tasks>
</total>
```

4.2 General Substructures

4.2.1 Stringlang

General string with language attribute.

Definition:

<[stringlang]>
attribute: (lang)
data: **string**
</[stringlang]>

Note that the name of the structure may vary.

Attribute of note:

Element	Type/description	Level
lang	lang: The language of the note.	4

4.2.2 Normstringlang

General normalized string with language attribute.

Definition:

<[normstringlang]>
attribute: (lang)
data: **normalizedString**
</[normstringlang]>

Note that the name of the structure may vary.

Attribute of note:

Element	Type/description	Level
lang	lang: The language of the note.	4

4.2.3 Object

Objects represents a file. An object may be included as a reference to an external file (using *externalfile*) or completely included in the GenXML file (using *originalfile* and *data*). If *object* is used as a reference to an external file, that external file should be present in the same folder as the GenXML-file.

Definition:

<object>
attribute: (lang?)
content: (objtype?, (externalfile | (originalfile, bin)), title, author?, owner?, note?, ext*)
</object>

Attribute of object:

Element	Type/description	Level
lang	lang: The language of the object information (and content if applicable).	4

Content of object:

Element	Type/description	Level
objtype	"unknown": The object type is unknown. This is the default value.	3
	"portrait": The object contains a single person, like a picture of a person, a movie of a person, or a recording of a person.	
	"group": The object contains a group of people, like a picture of a group, a movie of a group, or a recording of a group.	
	"location": Te object contains a location, meaning a picture or movie of a location.	
	"other": The object mainly doesn't contain people. For example the picture of a house, or the sound of a dog. This will usually be the correct type for source objects.	
externalfile	token: The name of the external file. No directory information may be included. All files should be exported to (or imported from) the same directory.	3
originalfile	token: The name of the original file.	3
bin	base64Binary: The object data base64-encoded.	3
title	normstringlang: Short description of the object. In case of pictures, this is the text that would normally be placed below the picture when printed.	3
author	normalizedString: Name of the photographer (personal name or institution) for pictures and movies. The author if the object is a document.	4
owner	normalizedString: Name of owner of original (personal name or institution). Useful for pictures only.	4
note	string: Note.	3
[ext]	any: Any simple or complex type of other namespaces	N/A

4.2.4 Address

The address structure represents an postal address of something, someone or where something happened. It is formatted as it would be for example on a mailing label.

The tag <lf/> is used to separate lines. Note that addresses often are limited to four lines (excluding the addressee). Of these, the third line is usually used for the city-name and postal code, while the fourth line is used for the name of the country. If, four example, the second line is not used, it should still be included (by an extra <lf/>, se the examples). There may or may not be an <lf/> after the last line. Importing applications should be aware though, that in some countries the address is written "upside-down". In such cases the country will be put in the first line, the city in the second etc.

An importing application that is not level 4 compliant, will usually ignore the tp-attribute.

Definition:

<address>
attributes: (lang?)
content: ((ap | lf)*, phone?, fax?, ext*)
</address>

Attributes of address:

Element	Type/description	Level
lang	lang: The language in which the address is written.	4

Content of address:

Element	Type/description	Level
ap	normalizedString: Address part. Includes the optional attribute *tp* (see definition below).	1/4
lf	no data: Line feed. Indicates that the following ap-fields belong to the next address line.	1
phone	normalizedString: Phone number at this address.	2
fax	normalizedString: Fax number at this address.	3
[ext]	any: Any simple or complex type of other namespaces	N/A

Possible values of the *tp* attribute of the *ap* field:

Element	Type/description	Level
tp	"name": The address part refers to the name of the house. This is only used if the house have a name and it is used as a part of the postal address.	4
	"street": The address part refers to a street name.	4
	"number": The address part refers to the number of the house. This is usually not included without the name of the street.	4
	"district": The address part refers to the local district. This is mainly used when there is no street name.	4
	"pobox": The address part refers to a postal office box. Both the number of P. O. box and the description of it should be included. (For example "P. O. Box 57".)	4
	"pcode": The postal code or ZIP code.	4
	"city": The address part refers to a town or city, or it may be the name of the nearest post office.	4
	"country": The address part refers to a country.	4
	"state": The address part refers to a state name. This is used only in USA, Canada and Australia.	4
	"other": The address part is of some other kind than those above.	4
	"unknown" (default): The address part is of an unknown type, or the exporting program does not support the above address part descriptions.	1

Examples of level 1 addresses:

```
<address>
   <ap>Øvre Strandgate 75</ap><lf/><lf/>
   <ap>4300 Stavanger</ap><lf/>
   <ap>Norway</ap>
</address>

<address>
   <ap>50 East North Temple Street</ap><lf/><lf/>
   <ap>Salt Lake City, UT 84150</ap><lf/>
   <ap>USA</ap>
</address>
```

Examples of level 4 addresses:

```
<address>
   <ap tp="name">Madsegården</ap><lf/>
   <ap tp="district">Brueland</ap><lf/>
   <ap tp="pcode">4300</ap>
   <ap tp="city">Sandnes</ap><lf/>
   <ap tp="country">Norway</ap>
</address>
```

4.2.5 Personalname

The *personalname* structure records the name of a person. The structure is used for two purposes:

1. In the *person* structure for storing the name used to identify the person in the database.
2. In the *alias* structure for storing the name found in a source. In many cultures in may be useful to normalize the name, while storing the name exactly as found in the source in the referred *excerpt* structure.

The name is split into name parts. Note that a name part may consist of more than one name. For example

```
<np tp="givn">Johan</np><np tp="givn">Henrik</np>
```

is equivalent with

```
<np tp="givn">Johan Henrik</np>
```

Name parts like articles and prepositions may be included with the name part it belongs to, or it may be regarded as a separate name part using tp="art". This should somehow be decided by the user who registers the name. One simple rule to decide this is how the user wants the name sorted. If you want the name "Godske von Ahlefeld" sorted by surname as "von Ahlefeld, Godske" then it should be recorded as

```
<np tp="givn">Godske</np>
<np tp="surn">von Ahlefeld</np>
```

If you want it sorted by surname as "Ahlefeld, Godske von" it should be recorded as

```
<np tp="givn">Godske</np>
<np tp="art">von</np>
<np tp="surn">Ahlefeld</np>
```

Definition:

<personalname>
  attribute: (lang?)
  content: (np+, maid?)
</personalname>

Attribute of personalname:

Element	Type/description	Level
lang	lang: The language in which the name is written.	4

Content of personalname:

Element	Type/description	Level
np	normalizedString: Name part. Requires the attribute tp (see below).	1

Possible values of the *tp* attribute of the *np* field:

Element	Type/description	Level
tp	"cogn": A cognomen or agnomen (nickname) given in addition to the other name(s) (like "the great" or "Germanicus").	4

	"surn": Surname. The surname, family name, clan name or similar inherited name of the person. Each person has normally only one surname. If the name is a surname, but not *the* surname of the person, the correct description is "midl" and not "surn". Also note that because a son has the same name as his father does not necessarily mean that the name is automatically inherited and may therefore not be a true surname but a cognomen, occupation name, locality name or similar.	1
	"pref": Title or prefix that is normally included as a part of the name, like "Sir".	4
	"givn": Given name. The given name of the person. The person may have several of these.	4
	"reln": Religious name. This is a name the person is given, or has taken, as a religious name and not as a normal given name.	4
	"nick": A nickname substituting the given name (like "Bill" instead of "William").	4
	"patr": Patronymic name. A name created from the person's father's name.	4
	"matr": Matronymic name. A name created from the person's mother's name.	4
	"tekn": Teknonymic name. A name created from one of the person's children's name.	4
	"art": Article. ("de", "von", "of" or similar)	4
	"occn": Occupation name.	4
	"ordi": Ordinal. Like "III" in "Charles III of France"	4
	"locn": Locality name (toponym).	4
	"midl": Middle name. Any kind of family name that is not the surname of this person, like the maiden name of a married woman. A middle name is a family name that may be the inheritable surname for other persons or families, but not for this person. This is not middle name in the American sense of the term. The Americans don't have middle names in the GenXML sense. If a person has two given names they should be coded as two "givn" and not as one "givn" and one "midl".	4
	"sufx": Suffix. ("sr.", "jr." or similar.)	4
	"oth": A name part of some other type than the ones above.	4
	"unkw": A name part which type is unknown.	1
maid	Maiden name, not to be confused by middle name or any other name parts above. The maiden name stored here is not really a part of the name but additional information. See examples below	4

Examples of level 1 names:

```
<personalname>
   <np tp="unkw">Fred</np>
   <np tp="surn">Lunde</np>
</personalname>

<personalname>
   <np tp="unkw">Gaius</np>
   <np tp="surn">Julius</np>
   <np tp="unkw">Caesar</np>
</personalname>
```

Examples of level 4 names:

```
<personalname>
   <np tp="givn">Billy</np>
   <np tp="cogn">the Kid</np>
</personalname>

<personalname>
   <np tp="givn">Godske</np>
   <np tp="art">von</np>
```

```
   <np tp="surn">Ahlefeld</np>
   <np tp="art">til</np>
   <np tp="locn">Bosse</np>
   <np tp="art">og</np>
   <np tp="locn">Lindau</np>
</personalname>

<personalname>
   <np tp="givn">Gaius</np>
   <np tp="surn">Julius</np>
   <np tp="cogn">Caesar</np>
</personalname>

<personalname>
   <np tp="nick">Bill</np>
   <np tp="surn">Clinton</np>
</personalname>
```

Note that although many of the Roman cognomens were inherited, they were not true surnames. They were not necessarily inherited by all family members. We may often see that the oldest son inherits the cognomen of his father, while the younger sons get a different cognomen.

In some cultures (for example Germany) it's common to write a married woman's name like "Hedvig Alvarstein, born Waaler". This indicates that Waaler is no longer a part of the name although it used to be. This should be encoded as:

```
<personalname>
   <np tp="givn">Hedvig</np>
   <np tp="surn">Alvarstein</np>
   <maid>Waaler</maid>
</personalname>
```

Please note that this is distinctly different from the case where a woman keeps her maiden name as a middle name, like "Hedvig Waaler Alvarstein" which should be encoded as:

```
<personalname>
   <np tp="givn">Hedvig</np>
   <np tp="midl">Waaler</np>
   <np tp="surn">Alvarstein</np>
</personalname>
```

4.2.6 Coords

This structure holds the geographical coordinates of a place name.

Definition:

<coords>
content: (lo, la)
</coords>

Content of coords:

Element	Type/description	Level
lo	dDD, dDDMM or dDDMMSS: Longitude of the place, in degrees (DD), minutes (MM) and seconds (SS). The prefix (d) must be either "W" or "E".	4
la	dDD, dDDMM or dDDMMSS: Latitude of the place, in degrees (DD), minutes (MM) and seconds (SS). The prefix (d) must be either "N" or "S".	4

4.2.7 Date

The date structure represents a specific date (known or unknown) when something happened, or it may represent a period of time.

Note that the original date phrase belongs in the *excerpt* structure. The *date* structure holds the dates that are to be presented in reports.

Definition:

<date>
content: ((exact | (begin, end) | (from, to?) | to | text), sortdate)
</date>

Content of date:

Element	Type/description	Level
exact	simpledate structure (see 4.2.8): An exact date, known or unknown.	1
begin	simpledate structure (see 4.2.8): 'begin' and 'end' represents an unknown date in a known date range. This kind of dating will normally be written as "between <begin> and <end>" in ordinary text.	3
end	simpledate structure (see 4.2.8): End date of a date range. See 'begin'.	3
from	simpledate structure (see 4.2.8): An open period starting at the specified point of time.	2
to	simpledate structure (see 4.2.8): An open period ending at the specified point of time.	2
text	normalizedString: Any date written as free-form text. Other fields should be used if possible.	2
sortdate	datetime: A normalized date used mainly for sorting and calculation purposes. This tag can only hold a single, exact date.	4

Examples:

```
<date>
   <exact>2001-12-06</exact>
</date>

<date>
   <begin>1904-05-00</begin>
   <end>1904-06-00</end>
</date>

<date>
   <from>1874-12-04</from>
   <to>1876-02-00</to>
</date>
```

4.2.8 Simpledate

Simpledate represents a single date. The format of the date itself is based on the XML Schema Specification's dateTime datatype. Please note that the time zone part of dateTime is not a part of *simpledate*. The reason for this is that in an GenXML event, the time zone is defined by the recorded place. If the place is unknown, so is the time zone.

Definition:

<[simpledate]>

attributes: (cal?, mod?, era?)
data: **date**
</[simpledate]>

Note that the name of the structure may vary.

Attributes of simpledate:

Element	Type/description	Level
cal	"chinese": It is a date in the Chinese calendar.	4
	"coptic": It is a date in the Coptic or Ethiopian calendar.	4
	"french": It is a date in the French revolutionary calendar.	4
	"gregorian": It is a date in the Gregorian calendar.	1
	"hebrew": It is a date in the Jewish calendar.	4
	"indian": It is a date in the Indian calendar.	4
	"islamic": It is a date in the Islamic calendar.	4
	"julian": It is a date in the Julian calendar.	3
	"unknown" (default): Unknown calendar.	3
mod	"about": The exact date is close to the given date.	2
	"after": The exact date is after the given date.	2
	"before": The exact date is before the given date.	2
	"estimated": The exact date is estimated as specified.	2
era	"after" (default) or "before": Indicates if the date is after or before the start of this era. Note that "before" is only defined for the Julian calendar and the proleptic Gregorian calendars (equals "before Christ").	3

Data of simpledate:

Element	Type/description	Level
date	This format of this field is a modified extension of the XML Schema Specification's dateTime datatype which in turn is based on ISO 8601. The general format is: YYYY[/ZZ][-MM[-DD[**T**tt:mm[:ss]]]] The date is represented by the year (0000 or greater) and the alternate year (00 or greater), the month (0 – 12 or 13 depending on the calendar used), and the day (0 – 31). Zero indicates that that part of the date is unknown. The number of digits is fixed. The alternate year (ZZ) is optional and is only used to show the possible date alternatives for dates using the Julian calendar. For example 1532/33-02-16 indicates that the date was in the year 1532 because New Year's Day was 25. March, but would have been in the year 1533 if New Year's Day was 1. January. tt:mm:ss represents the time (times, minutes and seconds), ranging from 00:00:00 to 24:00:00. Note that 2002-04-12T00:00:00 equals 2002-04-11T24:00:00. Since the time 00:00:00 does have a meaning, the time should be omitted if it is unknown. The number of digits for each part is always fixed, i. e. 3 July 1957 should be represented as 1957-07-03 and not 1957-7-3.	1/3/4

Examples: See the date structure.

4.3 Substructures of the Header Structure

4.3.1 Owner

The owner substructure contains information on the owner of the exported database.

Definition:

<owner>
content: (name, address?, phone?, email?, uri?)
</owner>

Content of owner:

Element	Type	Level
name	normalizedString: The name of the owner.	N/A
address	address structure (see section 4.2.4): The address of the owner.	N/A
phone	normalizedString: Phone number of the owner.	N/A
email	normalizedString: Email address of the owner.	N/A
uri	anyURI: Homepage of the owner.	N/A

4.4 Substructures of the Person Structure

4.4.1 Subpersons

The *subpersons* structure is used for combining two (and only two) persons that probably (but not necessarily) were the same individual. For more information, see the *person* structure.

Definition:

<subpersons>
attributes: (probability?)
content: (personref, personref, note?)
</subpersons>

Attributes of subperson:

Element	Type/description	Level
probability	int: The probability (in %) of the referred persons being one and the same.	4

Content of subperson:

Element	Type/description	Level
personref	ident: Pointer to a person.	4
note	stringlang (see section 4.2.1): A note. This is the description of the assumption made.	4

Example:

```
<subperson probability="50">
   <personref>1743</personref>
   <personref>1746</personref>
</subperson>
```

4.5 Substructures of the Assertion Structure

4.5.1 Alias

The alias structure stores a name of a person as found in a source. The user may want to normalize it, though, and refer to an *excerpt* (from the encapsulating *assertion* structure) with the exact spelling.

Definition:

<alias>
attribute: (negative?)
content: (personalname, personref, date?)
</alias>

Attributes of alias:

Element	Type/description	Level
negative	boolean (Default is "false"): Indicates "negative" information. If set to "true", this name was not used by the referred person.	4

Content of alias:

Element	Type/description	Level
personalname	personalname structure (see section 4.2.5): The name of the person as recorded in the referred source.	3
personref	personref structure (see section 4.5.6): Pointer to the owner of this name.	3
date	date structure (see section 4.2.7): The date or period when this name was used.	3

Example:

```
<alias>
   <personalname>
      <np tp="unkw">John</np>
      <np tp="surn">Smith</np>
   </personalname>
   <personref seq="2">476</personref>
   <date><exact>1778-05-14</exact></date>
</alias>
```

4.5.2 Relationship

The relationship structure represents a parent-child relationship. The father-child and mother-child relationships may (and should if both relationships come from the same source) be combined into a single relationship structure.

Note that GenXML does not guarantee the match between selected father/mother tag and their recorded sex, as this depends on the user who entered the data and the application which accepted the data.

Definition:

<relationship>
attribute: (negative?)
content: (relation, (father | mother | parent), child)
</relationship>

Attributes of relationship:

Element	Type/description	Level
negative	boolean (Default is "false"): Indicates "negative" information.	4

Content of relationship:

Element	Type/description	Level
relation	"biological": The relationship is biological.	1
	"adoptive": The relationship is through adoption.	3
	"foster": The relationship is through foster-parents.	3
	"other": The relation or association is of some other kind.	3

father	personref structure (see section 4.5.6): ID-number of the father in this relationship.	1
mother	personref structure (see section 4.5.6): ID-number of the mother in this relationship.	1
parent	personref structure (see section 4.5.6): ID-number of the parent (with unknown sex) in this relationship.	3
child	personref structure (see section 4.5.6): ID-number of the child in this relationship.	1

4.5.3 Attribute

The attribute structure represents an attribute or characteristic of a person. As opposed to events, attributes typically take place over a period of time. However, the most important difference between events and attributes is that attributes have a text of some sort. For example the attribute "hair colour" will need to store the actual hair colour, and the attribute "education" will need to store the kind of education and school.

The exact type of attribute is dependant on both class and description, and attributes of the same class with identical descriptions are of the same type.

Attribute corresponds to the Characteristic entity of the Gentech Genealogical Data Model (GDM).

When compared to GEDCOM 5.5's individual attribute structure, the class (textclass, numberclass or flagclass) corresponds to the attribute tag (for example EDUC, NCHI or OCCU), while *description* corresponds to the TYPE tag of the Event_Detail structure.

Definition:

<attribute>
attribute: (negative?)
content: (description?, ((textclass, text) | (numberclass, number, modifier?) | flagclass), personref, date?, place?,address?)
</attribute>

Attributes of attribute:

Element	Type/description	Level
negative	boolean (Default is "false"): Indicates "negative" information.	4

Content of attribute:

Element	Type/description	Level
description	normstringlang (see section 4.2.2): Short description of the kind of attribute.	3
textclass	"caste": The caste to which this person belonged.	3
	"education": Education. Note that the graduation is an event.	3
	"email": Email address.	3
	"idnumber": Any kind of national ID number, like social security number.	3
	"language": The language spoken by the person (for example native language).	3
	"nationality": The nationality of the person.	3
	"physical": Any kind of physical description of the person, for example eye or hair colour, race, height and weight. (Height and weight must be treated as text as they should include unit.)	3
	"property": Property owned by the person.	3
	"religion": The religion with which the person was affiliated.	3

	"title": A title, like a nobility title or similar.	3
	"work": Occupation.	1
	"other": Any other kind of text-based attribute.	3
text	normstringlang (see section 4.2.2): The details of the attribute.	1
numberclass	"age": The age of a person.	4
	"children": The total number of children, not the calculated sum of registered children.	4
	"marriages": The total number of marriages, not the calculated sum of registered marriages.	4
	"other": Any other kind of number-based attribute.	4
number	int: The details of this attribute.	4
modifier	"exact" (default): The number specified is exact.	4
	"greater": The number is greater than specified.	4
	"less": The number is less than specified.	4
flagclass	"ancestor": This person is an ancestor of some unspecified person.	4
	"descendant": This person is a descendant of some unspecified person.	4
	"living": This person is still living, or was living at the specified date.	4
	"other": Any other kind of flag-based attribute.	4
personref	personref structure (see section 4.5.6): ID-number of the person to which this attribute apply.	1
date	date structure (see section 4.2.7): The date or period during which the attribute existed or was valid.	3
place	ref: Points to a place structure (see section 4.1.8) representing the place where the attribute existed (if any).	3
address	address structure (see section 4.2.4): The postal address of the place where the attribute was (if any).	3

Examples:

```
<!--Person with id=P57 was a fisherman-->
<attribute>
   <description>occupation</description>
   <textclass>work</textclass>
   <text>fisherman</text>
   <personref>P57</personref>
</attribute>

<!--Person with id=P193 had at least 7 children-->
<attribute>
   <numberclass>children</numberclass>
   <number>6</number>
   <modifier>greater</modifier>
   <personref>P193</personref>
</attribute>

<!--Person with id=P20 was not living 4. December 1964-->
<attribute negative="true">
   <flagclass>living</flagclass>
   <personref>P20</personref>
   <date><exact>1964-12-04</exact></date>
</attribute>
```

4.5.4 Event

The event structure represents an event in the life of one or more persons. An event is something that happened at a certain moment – a certain day. There are however a few exceptions, like residence. ("Residence" is logically an attribute, but technically an event as it has no special data except the place, which all events and attributes may have.)

The event may include many participants that have different roles in the event. However one must distinguish between principal participants in the event and participants that have a subordinate role. In a christening event, the person that is born is the principal participant. The godfather, godmother, priest, and all other participants have a subordinate role. In a wedding there are two principal participants: the bride and the groom. In some events, for example the invasion of Normandy 5. June 1944, there are a very large number of principal participants. (In such "group"-events, talking of subordinate participants has little meaning.)

All events have at least one principal participant. However in some cases the principal participant may be unknown. Therefore the principal field is not mandatory.

Note that although the parents of a child do have a subordinate role in the birth event of the child, the parent-child relationship should not be recorded using this mechanism. Instead use the relationship structure (see section 4.5.2).

Definition:

<event>
attribute: (type, negative?)
content: (principal*, subordinate*, date?, place?, address?)
</event>

Attributes of event:

Element	Type/description	Level
type	ident: ID-number of the event type (see section 4.1.6) of which this event is an instance.	1/3
negative	boolean (Default is "false"): Indicates "negative" information.	4

Content of event:

Element	Type/description	Level
principal	personref structure (see section 4.5.7): Pointer to the principal person of this event	1
subordinate	personrefrole structure (see section 4.5.7): Pointer to a person that has a subordinate role in this event.	4
date	date structure (see section 4.2.7): The date when the event happened.	1
place	ref: Pointer to a place structure (see section 4.1.8) representing the place where the event happened.	1
address	address structure (see section 4.2.4): The address of the place where the event happened.	3

Examples:

```
<event type="5" pref="true">
   <principal>53</principal>
   <subordinate >71</subordinate>
   <subordinate>72</subordinate>
   <date><exact>1978-10-29</exact></date>
   <place><pnp tp="city">Oslo</pnp><pnp tp="country">Norway</pnp></place>
</event>
```

4.5.5 Info

The info structure represents general information that does not fit well into other assertion structures.

Definition:

<info>
content: (text, personref+)
</info>

Content of info:

Element	Type/description	Level
text	stringlang structure (see section 4.2.1): This is the actual information.	1
personref	personref: Pointer to the person of which this information apply.	¼

4.5.6 Personref

The personref structure represents a link between an assertion and a person. A sequential number may be supplied. This states the sequence of assertions for a person. The number should be unique for each person, but it is not required. The sequence of assertions with equal sequential numbers for the same person, must be regarded as not specified.

Definition:

<[personref]>
attribute: (pref?, seq?)
data: **personid**
</[personref]>

Note that the name of the structure may vary.

Attribute of personref:

Element	Type/description	Level
pref	boolean: (Default is "true".) States if the encapsulating assertion is the preferred assertion of all assertions of the same type. If this attribute is not set for any assertions of a given type, the first one of that type may be regarded as the preferred one. If more than one assertions of the same type have this attribute set to "true", the first of these may be regarded as the preferred one.	3
seq	int: Sequential number specifying the sequence of assertions for the referred person.	4

Data of personref:

Element	Type/description	Level
personid	ident: Pointer to the person that has a role in the related event.	1

Example:

```
<personref seq="4">P85</personref>
```

4.5.7 PersonrefRole

The *personrefrole* structure is a pointer to a person that has a subordinate role in the event of which *personrefrole* is a subject.

Definition:

<[personrefrole]>
contents: (personref, role)
</[personrefrole]>

Note that the name of the structure may vary.

Content of personrefrole:

Element	Type/description	Level
personref	personref structure (see section 4.5.6): Pointer to the person that has a role in the related event.	3
role	normstringlang structure (see section 4.2.2): The role of this person in the related event.	3

Example:

```
<subordinate>
   <personref seq="7">P316</personref>
   <role>witness</role>
</subordinate>
```

4.6 Substructures of the source structure

4.6.1 Repositoryref

Definition:

<repositoryref>
attributes: (ref)
contents: (callnumber?, note?)
</repositoryref>

Attributes of repository:

Element	Type/description	Level
ref	ident: Pointer to a repository.	3

Contents of repository:

Element	Type/description	Level
callnumber	normalizedString: The callnumber of the source in the refered repository.	3
note	stringlang (see section 4.2.1): Notes concerning the specific copy of the source in the referred repository.	3

5 GenXML levels

The features of GenXML are divided into levels to make it easier both for the application developers and the users to understand what the capability of a genealogy program is.

The general description of each level is as follows:

1	The basic information of a simple genealogy program
2	Approximately the same information as the most commonly used parts of Gedcom 5.5
3	All information that an advanced genealogy application should support
4	In addition to level 3 data, all data of minor importance must also be supported

Definitions:

GenXML level X includes all so called level X structures in addition to all level X-1 structures.

A GenXML file is said to be a **GenXML level X file** if the file does not include any structures of higher levels.

A program said to be **GenXML level X compliant**, should be able to read and write all GenXML level X files as well as files of lower levels. When importing a GenXML level X file and then exporting the same data, all data should be exported in the same structures as in the imported file.

Data loss should never occur, except when importing a level X+1 (or greater) file into a level X compliant program.

See chapter 4 for details on what level each kind of data belongs to. A summary is given below. Note that some fields are marked as Not Applicable (N/A). These fields hold non-genealogical information of minor importance. Each application, regardless of GenXML level compliance, may choose to ignore this information.

5.1 Level 1

Level 1 is the most basic level. The following structures must at least be understood by a program in order to be called "level 1 compliant". A program that is not level 1 compliant is not GenXML compliant at all.

- The file structure with all fields.
- The header structure with all fields and substructures. Note that not all fields are mandatory.
- The basic eventtype structure. Note that only eventtypes of class "birth", "death" and "marriage" may be supported. Event types of the same class may be regarded as the same event type. The roles field may only take values "1" (for classes birth and death) or "2" (for class marriage).
- The person structure.
- The assertion structure. Only the following substructures:

    - The basic relationship structure. Only biological relationships, and only one relationship for each child (i. e. no alternative parentage).
    - The attribute structure. Only one attribute structure of each class per person, and only textclass="work".
    - The basic event structure. Only one event of each event type per person. Note the limitations in the eventtype structure.
    - The info structure. Only one per person, and only one person per info structure.
- The total structure. Note that all fields are mandatory.
- All dates may consist of only a single, exact date of the gregorian calendar without the time or alternate year specified.

5.2 Level 2

A program must, in order to be called "level 2 compliant", be able to read and write the following data, in addition to level 1 data:

- The basic source structure.
- The basic excerpt structure.
- Additions to the header structure:
    - The language attribute.
- Additions to the eventtype structure:
    - The classes "baptism", "burial" and "divorce".
- Additions to the person structure:
    - The excerptref field.
- Additions to the assertion structure:
    - The excerptref field.
- Additions to the address structure:
    - The phone field.
- Additions to the date structures:
    - The modifiers "before", "after", "about" and "estimated".
    - Dates may be described by a string.
    - The date structure may include a period instead of an exact date.

5.3 Level 3

A program must, in order to be called "level 3 compliant", be able to read and write the following data, in addition to level 2 data:

- The repository structure.
- Additions to the excerpt structure:
    - The text, quality and note fields.
- Additions to the eventtype structure:
    - All classes must be supported.
    - The description field. I. e. at level 3 and 4, the description field is used to distinguish between event types of the same class. At level 1 and 2, the description field (which is mandatory) is merely a comment, and there is no more than one event type of each class. (The class is in fact taking the place of the event type for these levels.)
    - The gedomtag field.
    - The roles field may take other values than 1 and 2.
- Additions to the person structure:
    - Sex may take the value "unknown".
    - The note field.
- Additions to the assertion structure:
    - The alias substructure.

    - Additions to the relationship structure:
        - There may be unlimited relationship assertions per person.
        - Relationships of other types than "biological".
        - Relationships with one parent of unknown sex.
    - Additions to the attribute structure:
        - There may be unlimited attribute assertions per person.
        - The description field.
        - Attributes of all textclasses.
        - The date and place substructures.
    - Additions to the event structure:
        - There may be an unlimited number of events of each event type for each person.
    - Additions to the info structure:
        - Unlimited number of info structures per person.
    - Additions to the personref structure:
        - The pref attribute.
    - The note field.
- The task structure.
- The lang attribute of the repository, source, excerpt and person structures.
- The object substructure.
- Additions to the address structure:
    - The fax field.
- Additions to the date:
    - Calendar may be "julian" or "unknown" (as well as "gregorian").
    - Both "after" and "before" are possible values of the era-attribute.
    - May include alternate year.

5.4 Level 4

A program must, in order to be called "level 4 compliant", be able to read and write the following data, in addition to level 3 data:

- Additions to the source structure:
    - The class, kind and media attributes.
    - The isbn and issn fields.
    - Source hierarchy.
- Additions to the eventtype structure:
    - The note field.
- Additions to the person structure:
    - The subpersons structure.
- Additions to the assertion structure:
    - The datatype attribute.
    - The assertionref field.
    - The negative attribute of the alias, relationship, attribute and event structures.
    - Additions to the attribute structure:
        - Numerical attributes. (The numberclass, number and modifier fields.)
        - Flag attributes. (The flagclass field.)
    - Additions to the event structure:
        - The subordinate structure.
    - Additions to the info structure:
        - There may be more than one personref structures, i. e. several persons may share a common info structure.

    - The seq attribute of the personref structure.
- The objective structure.
- Additions to the task structure:
    - The objectiveref field.
- More advanced address structure.
- More advanced place structure. It may also include a coords structure.
- More advanced name structure.
- The lang attribute of the stringlang, normstringlang, address and object structures.
- Additions to the date structures:
    - All calendars.
    - Inclusion of the time in datetime strings.

6 How To …

6.1 … Use Source Hierarchies

A source may be split into a hierarchy of sources. The top-level source represents the whole source. The user and/or the application decide the exact split of sources into hierarchies, but some examples will be given here. Source hierarchies are necessary when it is unclear what to call a specific source or when the user wants to record details of specific parts of a source. For example do you record each volume in a series as a source, or all of the series as a single source? What if the series consist of several related series? And what if each volume in the series is published in separate parts over a period of time? Source hierarchies solve all these problems.

Note that the complete source hierarchy of a specific source may be very large. Usually only small parts of the tree are actually registered – the parts that are needed.

Only the top-level source may be linked to a repository, as all other source records are part of the top-level source and therefore is stored in the same place as the top-level source.

If for example a book has been published in two different editions and you want to register both editions, they should not be parts of the same source hierarchies, but should be regarded as two different sources.

6.1.1 Example 1: Church Records

A church register may be registered as a top-level source of class "book". The church register may consist of several parts. For example, there may be part for marriages. This may be registered as a second-level source of class "part". The sourceref tag of this source will point to the top-level source. This part will contain several entries, each describing a marriage. One, some, or all, of these may be registered, as third-level sources of class "document".

6.1.2 Example 2: Encyclopedia Britannica

The complete Encyclopedia Britannica may be registered as a top-level source of class "series". A specific volume may be registered as a second-level source of class "book", referring to the top-level source. A specific article in that volume may be registered as a third-level source of class "document".

6.1.3 Example 3: A Periodical

You want to store a reference to an article in The American Genealogist, volume 77 number 1 (January 2002). The complete journal may be registered as a top-level source of class "series". The volume may be registered as a second-level source of class "book". The specific issue may be registered as a third-level source of class "issue", and the article may then be registered as a fourth-level source of class "article".

6.2 … Use Multiple Person Records for the Same Individual

The person structure is a collection of assertions related to a specific individual. In some cases you may be unsure if two pieces of information deal with the same individual or with two different individuals. In such cases you may create two person structures, and create a third, super, person structure connecting the two previous ones.

If you later can prove that the two individuals really are the same, you may choose to merge the records, or to keep all the person structures to show your reasoning.

6.3 ... Build Up Your Reasoning Using Assertions

Assertions are normally based directly on source fragments. For example, an excerpt "Elspeth Sinclair, relict of William Halcro", dated 31. July 1619, may give two assertions: 1. a marriage event between Elisabeth Sinclair and William Halcro, and 2. a death event of William Halcro, dated before 31. July 1619.

In some cases it might be useful to base an assertion on other assertions instead of directly on source fragments. The new assertion does not have to be of the same type as the assertions on which it is based.

For example, in a document Ninian Neven is said to be the son of John Neven. Based on this you may create a relationship assertion. In the same document John Neven is said to be the husband of Katherine Mowat. Based on this you may create a marriage event. In another document Ninian Neven is said to be the nephew of Gilbert Mowat, son of Andrew Mowat. Based on this you may simply create an info assertion telling that Ninian was the nephew of Gilbert Mowat. Combining this information you will probably guess that Katherine Mowat was the sister of Gilbert and mother of Ninian, although you have no proof. You may create two new relationship assertions based directly on the source fragments. However a better solution would be to base both of the new assertions on the first two assertions, showing your chain of thought.

6.4 ... Store Adoptions

Adoptions consist of two parts: the adoption event and the relationship between the adoptive parents and the adopted child. A complete adoption will thus consist of three assertions: one for the event, one for the adoptive father – adopted child relationship, and one for the adoptive mother – adopted child relationship. The last two may be combined into one.

6.5 ... Make the Most of the Research Model

GenXML supports one research project per file, but may have several research objectives. Each research objective should be split up into research tasks, where one task is related to one and only one person and one and only one source.

A research objective, such as "Who is the father of John Smith?" may require look-ups in several sources for several persons, and thus in several repositories (libraries). All of these combinations is recorded and specified in research tasks. Each source may be linked to all the repositories (usually a subset) in which it exists. The researcher may thus get a list of all tasks that need to be done in a specific repository (library), independent of which research objectives they are part of.

A research objective is not completed until all of its tasks are completed or rejected.

The user may decide to keep or delete research objectives and their tasks when they are completed.