GenXML 2.0

Draft 6. February 2003

Acknowledgement
Thanks to the members of the GEDCOM-L and GenealogyXML mailing lists, who have given valuable feedback on the different GenXML drafts.

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.

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

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 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. It's updated data model is a purer evidence/conclusion model.

The major differences between GenXML 1.0 and 2.0 are:

- 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*.
- Both the *group* and *couple* structures have been eliminated. *Group* is replaced by the *assertion* substructure *groupevent*, while *couple* is unnecessary when the *coupleevent*s are directly related to *person*s.
- The *document* structure has been removed. The evidence model is not developed far enough to include such a structure. Some alternative ideas have been suggested. One is to include sources using a TEI-like syntaks. The other is to implement an unlimited tree of sources, like GDM. Both of these may be difficult to implement in a way, which also supports the simpler model, as Gedcom and now also GenXML does. This somewhat simpler model will probably meet the demand by the majority of users of genealogy software. As a result of this, the *sourceref* and *excerpt* structures have been merged.

2 Properties of GenXML

2.1 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.

2.2 Evidence and Conclutions

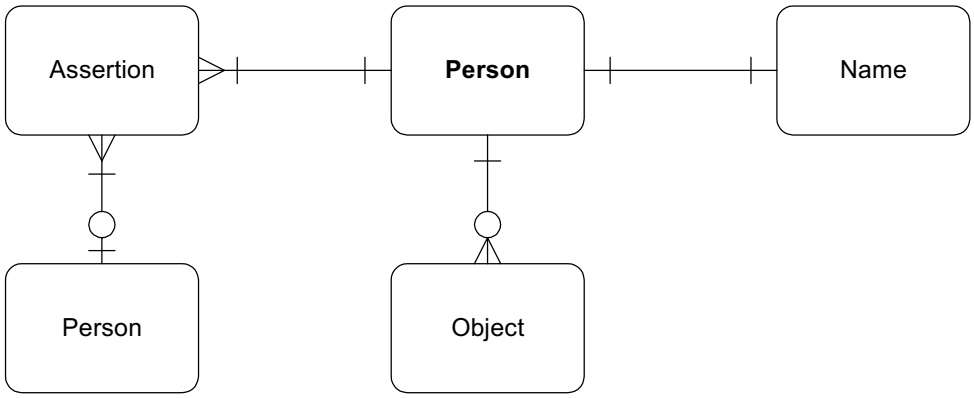
The GenXML format is based on an evidence/conclusion model.

2.2.1 The Evidence Submodel

The main structures of the evidence part is *source*, *document* 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.

GenXML has a simpler evidence model than the Gentech Data Model, which uses a more generalized source entity that may be combined in an unlimited number of levels. In GenXML the number of levels are limited to three: source, document and excerpt. This will probably be more than enough for most amateur genealogy programs and it will be much simpler to implement.

2.2.2 The Conclusion Submodel

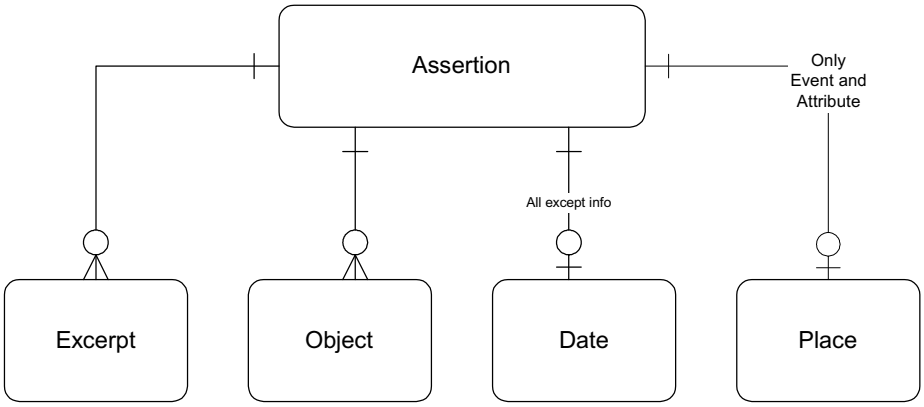


The conclusions are stored as *assertions*, and all assertions are linked together through the person structure which basically represents a single individual. There are several kinds of assertions:

- alias
- relationship
- attribute
- personevent
- coupleevent

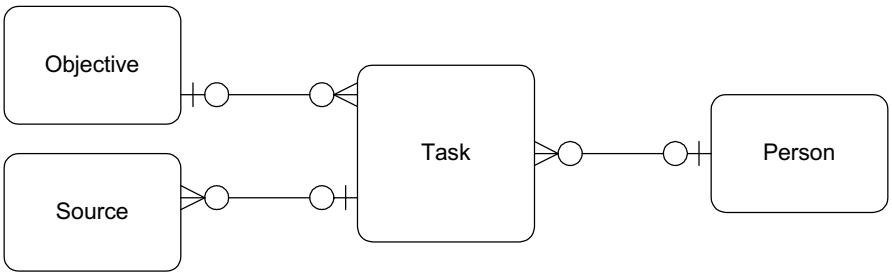
- groupevent
- info

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



2.3 The Research Process

The GenXML format also includes support for the research process. This includes the *objective* and *task* structures. The Gentech 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.

2.4 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 4.5.8).

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.

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', 'ref' and datetime are used as defined in the W3C XML Schema recommendation (see http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/).

Datatype	Definition
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 on the form YYYY-MM-DD tt:mm:ss (ISO 8601). Parts of it may be omitted such that the following forms may be used: YYYY YYYY-MM YYYY-MM-DD YYYY-MM-DD tt:mm YYYY-MM-DD tt:mm:ss
ident	A 32-bit signed integer greater than 0. This is used to identify main structures and has to be unique for each main structure of the same type.
int	A 32-bit signed integer.
lang	Identifies the language of the tag in which it is defined. Legal valuesare 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 like the datatype string, but without any carriage return (0x0D), line feed (0x0A) or tab (0x09) characters. So called whitespace characters (tab and space) must not be repeated.
ref	A 32-bit signed integer greater than 0. Used as a pointer to main structures, and should be equal to the id (of type ident) of that structure.
string	A string of unlimited length.

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.

Note that other character sets than ISO-8859-1 may be used, but will not necessarily be imported by any program. ISO-8859-1 should always be supported.

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*, document*, excerpt*, group*, assertion*, person*, objective*, task*, total)
</genxml>

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	4
assertion	assertion structure, see section 4.1.6	1
person	person structure, see section 4.1.7	1
objective	objective structure, see section 4.1.8	4
task	task structure, see 4.1.9	3
total	total structure, see section 4.1.10	1

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 a program only understands the 1.00-format, it should not import a 2.00-file – at least not without warning the user.

The level indicates what kind of data the importing program may expect. The file should not include data of higher levels than the given level. For example a level 2 compliant program (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 the value of this field must always be 200.	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 program is only level 2 compliant. This tells the importing program what it might expect.	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, language?, owner?, copyright?, note?, change?)
</header>

Content of header:

Element	Type/description	Level
exportingsystem	normalizedString: Name of the exporting system.	1
version	normalizedString: Version of the exporting system	1
exported	date: The date of the export.	1
language	lang: 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: Note.	N/A
change	datetime: The date and time of the last change of this database.	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?)
</repository>

Attributes of repository:

Element	Type/description	Level
id	ident: Unique identifier	3
lang	lang: 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	normalizedString: 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

Example:

```
<repository id="5" 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, kind?, media?, lang?)
content: (author, title, shorttitle, published?, (isbn | issn)?, repositoryref*, text?, object*, note?, change?)
</source>

Attributes of source:

Element	Type/description	Level
id	ident: Unique identifier	2
kind	"original": This is an 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	"printed": This is a printed source.	4
	"handwritten": This source is handwritten.	4
	"fiche": This source is published on microfiche.	4

	"microfilm": This source is published on microfilm.	4
	"photo": This source exists as a photography.	4
	"inscription": This source is an inscription, for example on a tombstone.	4
	"digital": This source exists as a digital file.	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	lang: 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
text	string: Text from the source.	2
object	object structure (see section 4.2.3): An object related to the document. (For example a picture of the document.)	4
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

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?, sourceid, note?, change?)
</excerpt>

Attributes of excerpt:

Element	Type/description	Level
id	ident: Unique identifier	2
lang	lang: The language of the excerpt.	4

Content of excerpt:

Element	Type/description	Level
text	string: Text extracted from the referred document.	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
sourceid	ref: Pointer to a source structure.	2

note	stringlang structure (see section 4.2.1): Note.	4
change	datetime: The date and time of the last change of this excerpt information.	N/A

The conversion between "quality" and Gedcom QUAY is recommended to be done according to the following table:

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%

4.1.6 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 | personevent | coupleevent | groupevent
info), (excerptref* | (assertionref | assertionref)), object*, note?, change?)
</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 couple information.	3

Content of assertion:

Element	Type/description	Level
alias	alias structure (see section 4.5.2): The assertion is a name alias.	
relationship	relationship structure (see section 4.5.3): The assertion is a relationship (a parent-child relationship or other relationship).	1
attribute	attribute structure (see section 4.5.4): The assertion represents an attribute of the associated person(s).	
personevent	personevent structure (see section 4.5.5): The assertion represents a personal event (like birth, death and graduation).	1
coupleevent	coupleevent structure (see section 4.5.6): The assertion represents an event involving two principal persons (like engagement, marriage and devorce).	1
groupevent	groupevent structure (see section 4.5.7): The assertion represents an event involving a group of people. Such an event may be the invasion of Normandy in 6. June 1944.	
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	1

	of the other assertion structures.	
excerptref	id: Pointer to an excerpt structure on which this assertion is based.	2
assertionref	id: Pointer to an assertion on which this assertion is based. An assertion may be based on two other assertions instead of an excerpt.	
object	object structure (see 4.2.3): Objects related to this couple.	3
note	stringlang structure (see section 4.2.1): Note.	2
change	datetime: The date and time of the last change of this couple information.	N/A

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: (name, (subpersons? | (assertionrefs?, children?, excerptref*, object*)), note?, change?)
</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
name	name structure (see section 4.2.6): 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 to persons into one.	4
assertionrefs	data structure (see section 4.4.2): The data structure includes all events, attributes, names, associations and comments.	1
children	children structure (see section 4.4.3): List of children.	1
excerptref	id: Pointer to an excerpt structure holding information which source and where in that source information on this person was found.	2
object	object structure (see section 4.2.3): External files related to this person.	3
note	stringlang structure (see section 4.2.1): Notes made by the researcher that does not fit in anywhere else. (The document and excerpt structures are preferred.) Comments that are meant to be printed on reports (for example the life of the person) should be stored in the info structure.	2
change	datetime: The date and time of the last change of this person information.	N/A

4.1.8 Objective

The objective structure represents a research objective. A research objective consists of one or more research tasks (see section **Feil! Fant ikke referansekilden.**). The research objective is split into one task for each affected person, couple or group.

Note that the research objective may be seen as having a status, like the research task. The objective's status equals the lowest status of its tasks. This may easily be calculated by a program. The research objective is not completed until all of its tasks are completed.

The task substructure 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, also the task structure is implemented as a main structure.

Definition:

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

4.1.9 Task

The task structure represents a research task. A research task may be related to a specific group, couple or person, and to one or more sources. It may (and should) also be part of a research objective.

Definition:

<task>
attribute: (id, lang?)

content: (title, problem, solution?, objective?, (group | couple | person)?, source?, status, statusdate?, change?)
</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
objective	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
group	ref: Pointer to a group structure.	3
person	ref: Pointer to a person structure.	3
source	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

4.1.10 Total

The TOTAL structure is always the last one in the file and should 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, assertions, persons, 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
excerps	int: The total number of excerpt structures in the file.	1
assertions	int: The total number of assertion structures in the file.	1
persons	int: The total number of person 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>
    <assertions>49</assertions>
    <persons>124</persons>
    <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: ((externalfile | (originalfile, bin)), title, note?)

</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
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	normalizedString: Short description of the object.	3
note	string: Note.	3

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 programs should be aware though, that in some countries the address is written "upside-down". It that cases the country will be put in the first line, the city in the second etc.

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

Definition:

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

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 Place

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

Definition:

<place>
attribute: (lang?)
content: (prefix?, pnp+, date?, coords?)
</place>

Attribute of place:

Element	Type/description	Level
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
pnp	normalizedString: Place name part. Includes the optional attribute *tp* (see definition below).	1/4
date	date structure (see section 4.2.8): The period when this place name was valid.	4
coords	coords structure (see 4.2.7): The geographical coordinates of the place.	4

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

Element	Type/description	Level
tp	"continent": The place name part refers to a continent or part of the world.	4
	"country": The place name part refers to a country.	4
	"state": The place name part 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 part 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 part refers to a town or city.	4
	"muni": The place name part refers to a municipality.	4
	"citypart": The place name part refers to a part of a town or city.	4
	"building": The place name part refers to a building or a group of buildings, like a church or monastery.	4
	"diocese": The place name part refers to an ecclesiastical district under the jurisdiction of a bishop.	4
	"deanery": The place name part refers to a district under the jurisdiction of a dean, part of a diocese.	4
	"parish": The place name part refers to an ecclesiastical district with its own church.	4
	"farm": The place name part refers to a farm or estate.	4
	"ocean": The place name part refers to an ocean or sea of salt water.	4
	"lake": The place name part refers to a lake.	4
	"mountain": The place name part refers to a mountain.	4
	"cemetery": The place name part refers to a cemetery.	4
	"other": The place name part refers to an other kind of geographical area or jurisdiction than the one above.	4
	"unknown" (default): The place name part is of an unknown type or the exporting program does not support the above place name part descriptions.	1

Example of level 1 place:

```
<place>
   <pnp>Balquholly Castle</pnp><pnp>Aberdeenshire</pnp><pnp>Scotland</pnp>
</place>
```

Example of level 4 place:

```
<place>
   <pref>near</pref>
   <pnp tp="building">Balquholly Castle</pnp>
   <pnp tp="county">Aberdeenshire</pnp>
   <pnp tp="country">Scotland</pnp>
</place>
```

4.2.6 Name

The name structure records the name by which the user wants to identify the person and not necessarily as found in a source. Names as found in a source is stored in the alias structure (see section 4.5.2).

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:

<name>
attribute: (lang?)
content: (np+)
</name>

Attribute of name:

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

Content of name:

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 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
	"locn": Locality name.	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

Examples of level 1 names:

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

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

Examples of level 4 names:

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

<name>
   <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>
</name>

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

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

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.

4.2.7 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

4.2.8 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 | (from, to) | text)
</date>

Content of date:

Element	Type/description	Level
exact	simpledate structure (see 4.2.9): An exact date, known or unknown.	1
begin	simpledate structure (see 4.2.9): '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 4.2.9): End date of a date range. See 'begin'.	3
from	simpledate structure (see 4.2.9): 'from' and 'to' represents periods of time. They may be used alone or together.	2
to	simpledate structure (see 4.2.9): End date of date period. See 'from'.	2
text	normalizedString: Any date written as free-form text. Other fields should be used if possible.	2

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>
```

4.2.9 Simpledate

Simpledate represents a single date.

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.	3
	"coptic": It is a date in the Coptic or Ethiopian calendar.	3
	"french": It is a date in the French revolutionary calendar.	3
	"gregorian": It is a date in the Gregorian calendar.	1
	"hebrew": It is a date in the Jewish calendar.	3
	"indian": It is a date in the Indian calendar.	3
	"islamic": It is a date in the Islamic calendar.	3
	"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 (equals "before Christ").	3

Data of simpledate:

Element	Type/description	Level
date	This format of this field is an extension of ISO 8601 and the general format is: YYYY[/ZZ][-MM[-DD[tt:mm[:ss]]]] The date is represented by the year (0000 or greater) and the alternate year (0000 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	1/3/4

	00:00:00 to 24:00:00. Note that 2002-04-12 00:00:00 equals 2002-04-11 24:00:00. 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.	

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
email	normalizedString: Email address of the owner.	N/A
uri	normalizedString: 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)
</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	ref: 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>
```

4.4.2 Assertionrefs

The assertionrefs structure encapsulates all assertion references.

Definition:

<assertionrefs>
attributes: (order?)
content: (assertionref*)
</assertionrefs>

Attributes of assertionrefs:

Element	Type/description	Level
Order	"none" (default): No particular order.	4
	"asc": The assumed order of the sub-structures. Oldest data first.	4
	"desc": The assumed order of the sub-structures. Youngest data first.	4
	"other": The substructures are sorted in some other way.	4

Content of assertionrefs:

Element	Type/description	Level
assertionref	ref: Pointer to an assertion structure.	1

4.4.3 Children

The children structure includes a list of all the children of a person. The real link between a parent and a child is not in the children structure, but in an association structure of the child. The children structure is nevertheless compulsory, because of easier manipulation and readability of the file. In addition, this structure includes the order attribute, making it possible to preserve the order of the children through an export and import, whatever order that is.

The order may not necessarily be the correct one. A statement from some source that "John was the third son of Patrick" is an excerpt on which an association structure (of John) should be based. The children structure of Patrick should, however, be based on that and other such statements, to make it as close to the truth as possible.

Definition:

<children>
attribute: (order?, nchildren?)
content: (child*)
</children>

Attributes of children:

Element	Type/description	Level
order	"none" (default): The child fields are in no particular order.	4
	"asc": The child fields are placed in the assumed order of birt. The oldest child first.	4
	"desc": The child fields are placed in the assumed order of birt. The youngest first.	4

	"other": The order of the child fields are of some other kind.	4
nchildren	int: The total number of children, if known. This is not necessarily the number of child fields. If the number of children are unknown, this attribute should not be present.	3

Content of children:

Element	Type/description	Level
child	ref: Pointer to person-structure assumed to be representing a child of this person. An association-structure need to be present for the child, and will keep all data for the relationship.	1

4.5 Substructures of the Assertion Structure

4.5.1 Subordinate

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

Definition:

<subordinate>
attribute: (personid)
content: (role?)
</subordinate>

Attributes of subordinate:

Element	Type/description	Level
personid	ref: Pointer to the person that has a subordinate role in the related event.	3

Content of subordinate:

Element	Type/description	Level
role	normstringlang (see section 4.2.2): The role of this person in the related event.	3

4.5.2 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* with the exact spelling.

Definition:

<alias>
attribute: (negative?)
content: (name, principal, date?)
</alias>

Attributes of alias:

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

Content of alias:

Element	Type/description	Level
name	name structure (see section 4.2.6): The name of the person as	2

	recorded in the referred source.	
principal	id: Pointer to the owner of this name.	2
date	date structure (see section 4.2.8): The date or period when this name was used.	3

4.5.3 Relationship

The relationship structure may represent a parent-child relationship, or any other relationship (using class="other"). Note that each parent-child relationship also requires a corresponding child field in the parent's children structure.

Definition:

<relationship>
attribute: (class, pref?, negative?)
content: ((description | relation), principal, relative, date?)
</relationship>

Attributes of relationship:

Element	Type/description	Level
class	"father": The referred person is the father of this person. The referred person must have a corresponding child field.	1
	"mother": The referred person is the mother of this person. The referred person must have a corresponding child field.	1
	"other": This is not a parent-child relationship.	3
pref	boolean: (Default is "true".) States if this is the preferred structure of all association structures of the same class ("true"). If this attribute is not set for any structures of the class, the first one may be regarded as the preferred one. If more than one associationof the same class have this attribute set to "true", the first of these may be regarded as the preferred one.	2/3
negative	boolean (Default is "false"): Indicates "negative" information.	4

Content of relationship:

Element	Type/description	Level
description	normstringlang (see section 4.2.2): Short description of the kind of association.	3
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
principal	id: Pointer to the principal person of this relationship.	1
relative	id: Pointer to the relative of the principal person of this relationship. If for example the relationship is of class 'father', then the principal person is the child and the relative is the father.	1
date	date structure (see section 4.2.8): The date when this association was established, or the period during which this association was valid. For example this may be the date of an adoption. There is no point in storing the birth date of a person here.	3

4.5.4 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.

Like events, attributes are of the same type only when both class and description are identical.

Attribute corresponds to the Characteristic entity of GDM.

Definition:

<attribute>
attribute: (class, pref?, negative?)
content: (description, details, personid, date?, place?)
</attribute>

Attributes of attribute:

Element	Type/description	Level
class	"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.	3
	"property": Property owned by the person.	3
	"religion": The religion with which the person was affiliated.	3
	"residence": A place where the person lived for some time.	3
	"title": A title, like a nobility title or similar.	3
	"work": Occupation.	1
	"other": Any other kind of attribute.	3
pref	boolean: (Default is "true".) States if this is the preferred structure of all attribute structures of the same kind ("true"). If this attribute is not set for any structures of the kind, the first one may be regarded as the preferred one. If more than one attribute of the same kind have this attribute set to "true", the first of these may be regarded as the preferred one.	2/3
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
details	normstringlang (see section 4.2.2): The details of the attribute.	1
personid	id: Pointer to the person to which this attribute apply.	1
date	date (see section 4.2.8): The date or period during which the attribute existed or was valid.	3
place	place (see section 4.2.5): The place where the attribute was (if any).	3

4.5.5 Personevent

The event structure represents an event in the life of the person. An event is something that happened at a certain moment – a certain day.

The event may include many participants that have different roles in the event. However in a *personevent* there may only be one principal participant. In a birth event, the person that is born is the principal. An unlimited number of persons may have subordinate roles in the event. Note that although the parents of a child may 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 4.5.3).

Events of the same class and with the same description should be regarded as being of the same type by the importing program. There are a limited number of different *classes* of events. There is, however, no limit of possible event-types, as these as differentiated by the description. The purpose of the class-field is to make it easy for programs to recognize certain events that may have a special meaning for example in reports.

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

Definition:

<personevent>
attribute: (class, pref?, negative?)
content: (description, principal, subordinate*, date?, place?)
</personevent>

Attributes of personevent:

Element	Type/description	Level
class	"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
	"election": The event of being elected or similar.	3
	"emigration": The event of leaving the place where one live.	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
	"immigration": The event of entering a new locality with the intention of residing there.	3
	"internment": The event of being interned, or similar.	3
	"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
	"retirement": Any kind of retirement.	3
	"other": Any other kind of events.	3
pref	boolean: (Default is "true".) States if this is the preferred structure of all personevent structures of the same kind ("true"). If this attribute is not set for any structures of the kind, the first one may be regarded as the preferred one. If more than one structures of the same kind have this attribute set to "true", the first of these may be regarded as the preferred one.	2/3
negative	boolean (Default is "false"): Indicates "negative" information.	4

Content of personevent:

Element	Type/description	Level
description	normstringlang (see section 4.2.2): Short description of the event.	3
principal	id: Pointer to the principal person of this event	1

subordinate	subordinate structure (see section 4.5.1): Pointer to a person that has a subordinate role in this event.	4
date	date structure (see section 4.2.8): The date when the event happened.	1
place	place (see section 4.2.5): The place where the event happened.	1

Examples:

```
<event class="baptism" pref="true">
   <description>christening</description>
   <principal>53</principal>
   <date><exact>1978-10-29</exact></date>
</event>

<event class="birth">
   <description>stillborn</description>
   <principal>1762</principal>
   <date><exact>1954-09-04</exact></date>
   <place><pnp tp="city">Oslo</pnp><pnp tp="country">Norway</pnp></place>
<event>
```

4.5.6 Coupleevent

Coupleevents represent events with two principal participants (although one of them may be unknown). Typical examples are engagement, marriage and divorce.

Events of the same class and with the same description should be regarded as being of the same type by the importing program. There are a limited number of different *classes* of events. There is, however, no limit of possible event-types, as these as differentiated by the description. The purpose of the class-field is to make it easy for programs to recognize certain events that may have a special meaning for example in reports.

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

Definition:

<coupleevent>
attribute: (class, pref?, negative?)
content: (description, principal, principal?, subordinate*, date?, place?)
</coupleevent>

Attributes of coupleevent:

Element	Type/description	Level
class	"annulment": The annulment of a marriage.	3
	"divfiling": The event of filing for a divorce by a spouse.	3
	"divorce": The event of dissolving a marriage.	2
	"marriage": The event of uniting two persons in marriage or similar union.	1
	"other": Any other kind of events.	3
pref	boolean: (Default is "true".) States if this is the preferred structure of all personevent structures of the same kind ("true"). If this attribute is not set for any structures of the kind, the first one may be regarded as the preferred one. If more than one structures of the same kind have this attribute set to "true", the first of these may be regarded as the preferred one.	2/3
negative	boolean (Default is "false"): Indicates "negative" information.	4

Content of coupleevent:

Element	Type/description	Level
description	normstringlang (see section 4.2.2): Short description of the event.	3
principal	id: Pointer to the principal person of this event.	1
subordinate	subordinate structure (see section 4.5.1): Pointer to a person that has a subordinate role in this event.	4
date	date structure (see section 4.2.8): The date when the event happened.	1
place	place (see section 4.2.5): The place where the event happened.	1

Example:

```
<coupleevent class="marriage" pref="true">
   <description>married</description>
   <principal>13</principal>
   <principal>14</principal>
   <subordinate personid="78">
      <role>best man</role>
   </subordinate>
   <subordinate personid="39">
      <role>maid of honour</role>
   </subordinate>
   <date><exact>1837-04-14</exact></date>
</coupleevent>
```

4.5.7 Groupevent

The *groupevent* structure represents an event valid for a group of people, which may however have different roles in the event. Note that as there are no principal participant in a groupevent, as opposed to *coupleevent* and *personevent*.

Definition:

<groupevent>
attribute: (negative?)
content: (text, subordinate+, date?)
</groupevent>

Attributes of groupevent:

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

Content of groupevent:

Element	Type/description	Level
text	normstringlang (see section 4.2.2): Description of the groupevent. Note that there are no event classes or event types for groupevents.	3
subordinate	subordinate structure (see section 4.5.1): Includes a pointer to a person to which this event is related. Also includes a description of which role the person had in the event.	3
date	date structure (see section 4.2.8): The date from when this person was a member of this group, or the period when this person was a member of this group.	3

4.5.8 Info

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

Definition:

<info>
content: (text, personid)
</info>

Content of info:

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

4.6 Substructures of the source structure

4.6.1 Repositoryref

Definition:

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

Attributes of repository:

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

Data 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

NOTE: THIS CHAPTER IS NOT YET UPDATED.

The features of GenXML is devided into levels to make it more easy both for the program 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 program 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 compulsory but belongs to a level higher than one. Such fields must always be exported, even if they are not imported by the same program.

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 compulsory.
- The couple structure.
- The person structure. Note that not all substructures are compulsory.
    - Only one info substructure.
    - Only one association structure of each class.
    - Only one personevent structure of each class. Only "birth" and "death" events may be supported.

    - Only one attribute structure of each class, and only class="work".
- The total structure. Note that all fields are compulsory.
- All dates may consist of only a single, exact date of the gregorian calendar without the time or alternate year specified.
- The description field of the association, personevent, attribute and coupleevent must always be exported, but may not be imported.

The substructures note, sourceref, object and todo are excluded from level 1.

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:

- Additions to the header structure:
    - The language attribute.
- The source structure and all legal references to it.
- Additions to the couple structure:
    - Additional notes (note).
- Additions to the person structure:
    - Additions to the data substructure:
        - The attribute "order", indicating the order of the substructure.
        - One alias substructure.
        - Personevent structures of the classes "baptism" and "burial".
        - When reading a file with a person with more than one alias structure or more than one association, personevent or attribute of the same class, the program must choose the first one with pref="true". If none of the said structures has this attribute, the first one must be chosen. For example if a person in a file has two personevents with class="baptism", the first with pref="false" and the second with pref="true", the second must be imported if only one of them is to be imported.
    - Additional notes (note).
- Additions to the date structure:
    - 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.
- The task structure.
- The couple structure: The number of children (nchildren).
- Additions to the person structure:
    - Possibility of sex="unknown".
    - Additions to the data substructure:
        - The groupref structure.
        - More than one alias substructure.
        - More than one info substructure.
        - More than one association of the same class. The "class" attribute may take the value "other". The "relation" field may take the values "adoptive", "foster" or "other", as well as "biological" .

            - More than one personevent of the same kind. All classes of personevents must be supported, including "other".
            - More than one attribute of the same class. All attribute-classes must be supported. The attribute structures may contain dates.
        - Additions to the children substructure:
            - The nchildren attribute.
        - Additions to the couples substructure:
            - The nmarriages attribute.
- The object substructure.
- Dates:
    - Calendar may be "julian" or "unknown".
    - Both "after" and "before" are possible values of the era-attribute.
    - May include alternate year.
- The pref attribute of the alias, personevent, attribute and coupleevent substructures.
- The description field of the association, personevent, attribute and coupleevent must be imported as well as exported.
- Additional note may be added to the alias, association, personevent, attribute, info and couplevent structures.
- The quality attribute of the sourceref structure.
- The lang attribute of the repository, source, couple and person structures.

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:

- The excerpt structure.
- The document structure.
- The objective structure.
- The group structure.
- Additions to the person structure:
    - The subpersons substructure.
    - Additions to the child substructure:
        - The order attribute.
    - Additions to the name substructure:
        - Names may be stored in a more advanced structure (see section 4.2.6).
- The datatype attribute may be added to the groupref, alias, association, personevent, attribute, info and coupleevent substructures.
- The place structure may be more advanced. It may also include a coords structure.
- The lang attribute of the note, object and place structures.
- Dates may include the time.

A The Future Development of GenXML

GenXML is currently being developed by me, Christoffer Owe. Ideally a replacement of Gedcom should be developed by some kind of organization. Unfortunately no organization has shown itself capable of doing this task yet. However if so happens I would be happy to hand over the responsibility to that organization. It would be a condition that GenXML remains an open standard and that the organization understands the ideas underlying GenXML.

I don't claim GenXML to be the perfect format. But in my opinion this is a great step ahead of Gedcom. Several parts of GenXML may be enhanced in a newer versions, but it must always be relatively easy to convert data from an old format (including Gedcom) to a new. If not, the new format will never gain popularity.

Below is a discussion on some parts that may be enhanced later.

A.1 Sources

The Gentech Data Model (GDM) has an evidence model mainly based upon a general source entity. This may be nested in several levels. GenXML has a simpler model, like Gedcom, using only two such levels, named *source* and *excerpt*. Should GenXML implement the GDM evidence model?

There is a problem that GDM doesn't solve. Suppose you type in a complete document with several pieces of information. This will be stored in a document structure. When using its pieces of information you might want to refer to them directly instead of retyping them into excerpt structures.

A possible way of doing this in a later version of GenXML may be to add a special syntax for coding documents (or text in general). Perhaps TEI (Text Encoding Initiative) will be the best format for this. The following questions need to be answered:

- Should the text encoding syntax be a part of GenXML or should it be an extension?
- If using an existing standard such as TEI – what should we do when new versions of the standard appear? Stay with the old one?
- How may we create links from for example events to parts of a document?
- How may such a model be used for downwards compatibility (with the Gedcom data model)?

Another solution could be to include templates for special documents in a similar way as GeniML does.

A.2 Modularity

It has been suggested that GenXML may be a modular format, where new parts may be added when needed. Ordinary genealogy programs probably have no need for this. It may, however, add the possibility of extending GenXML for use by more specialized programs for professional use. Some questions need to be answered:

- How should this modularity be legally defined in an XML schema? (Preferably namespaces must somehow be used.)
- What parts of GenXML should be modularized?
- What should a program do when importing a file including an unknown schema?

A likely solution is to modularize the place, address, and name structures by creating separate schemas for them. Each GenXML file may then select to use these schemas or choose alternative ones.