The DeadEnds Data Model

Version of September 2014
Thomas T. Wetmore IV

Introduction

This document describes the data model used by DeadEnds, a set of genealogy software programs. The model defines a set of object types (*classes*) that represent genealogical entities. Each DeadEnds objects is an instance of one of the classes, classes and is structured as hierarchical trees built from nodes of text-based information. Each node has a *tag* to identify its type, and *content* to hold its value. The content can be a text string and/or a list of nodes one level deeper the tree. Each node is best thought of as a *property* of the node one level above it in the tree. DeadEnds records are objects with unique identifiers that allow them to be referred to by other objects. The *root tag* of a record indicates its type. There are a number of record types in the model, and they are defined below.

The model is not abstract. Objects are read and written from files in the model format described here. DeadEnds databases are made up of the same objects formatted in the same way. There are different ways to represent text-based trees. The most general method in use today is XML. The most common used by genealogy programs is Gedcom. The specifications below use a notation somewhat simpler than XML and Gedcom. It is used as the format for database records and external DeadEnds files. DeadEnds programs can also import and export data in Gedcom and XML formats.

In the specifications below each node is structured by a few rules:

node :: *tag* **:** *content*

tag :: IDENTIFIER

content :: STRING? (**[** *node* (**;** node)* **]**)?

In these *rule definitions*, the term on the left of the double colon is defined by the expression on the right side. The first rule states that a *node* is a *tag* followed by a colon followed by *content*. The second rule states that a *tag* is an IDENTIFER. The third rule states that *content* is an optional STRING followed by an optional list of nodes between square brackets. The ? notation means 0 or 1 (optional) and * means any number including zero. The semicolon and square brackets in bold face are literals (they exist in the data). An IDENTIFIER is a short string of letters. A STRING is a Unicode string, which must be quoted only if it starts or ends with whitespace or contains any : or ; or [or] characters. The parentheses are used to group elements in rules.

An example text-based tree built from these rules is:

person : [id: abcd234fed5; name: Thomas Trask /Wetmore/ IV; gender: M]

This could be expressed in XML as:

<person id=" abcd234fed5">
<name>Thomas Trask /Wetmore/ IV</name>
< gender >M</ gender >

</person>

It could be expressed in Gedcom as:

0 @ abcd234fed5@ INDI
1 NAME Thomas Trask /Wetmore/ IV
1 SEX M

RecordSet Rule

The top level rule defines a set of DeadEnds records:

recordSet :: (*sourceRecord* | *eventRecord* | *personRecord* | *placeRecord* | *noteRecord* | *groupRecord* | *entityRecord* | *urlRecord* | ...)*

The vertical bar in this rule indicates alternatives. So this rule states that a recordSet is a sequence of zero or more records of any of the types defined below and in any order.

Source Records

Source records represent sources of genealogical evidence. Sources can be arranged in hierarchies.

sourceRecord :: **source : [** *id sourceContent* **]**

> All records have this basic structure, a root node with tag specifying the record type (here **source**) and a value consisting of an id node and the rest of the content.

id :: **id :** UUID

> Every record and record reference has an **id** node whose value is a UUID, a universally unique identifier. A UUID is a Unicode string that represents a 128 bit binary value. DeadEnds currently uses a custom string format for UUIDs that is 22 8-bit characters in length.

sourceContent :: *sourceType sourceAttr** *urlRef** *note** *noteRef** *sourceRef*?

> Source content starts with a type attribute followed by source attributes. It may also contain URL references, notes, note references, and an optional source reference to refer to the source that contains this source. See the sections that define the Note and Source records for the definitions of *noteRef* and *SourceRef*. These references can either refer to separate note and source records, or they can contain the note and source information *in situ*.

sourceType :: **type :** (**archive** | **library** | **townhall** | **courthouse** | **book** | **register** | ... | *userDefSourceType*) **;**

> The source type gives the type of the source. Since DeadEnds is a private format I make up new source type value strings as I need them.

sourceAttr :: *attribute*

> These are attributes of the source. Example attribute tags include **title**, **author**, **page**, **publicationDate**, **publicationPlace**, and **volume**. I make them up as I need them.

References to source records are found in records that refer to source records:

sourceRef :: **source :** (**[** *id sourceRefAttr** **]** | *sourceContent*)

> A source record may contain a source reference that refers to the source that contains this source.

sourceRefAttr :: *attribute*

These are attributes of the source referred to.

Event Records

Event records represent events in the lives of persons. Events involve persons as role players. They may serve to establish or change relationships between persons. Event records may represent events extracted directly from evidence, or they may represent hypotheses or conclusions on the part of the researcher.

eventRecord :: **event : [** *id eventContent* **]**

eventContent :: *eventType date*? *placeRef*? *personRoleRef** *eventAttr** *eventRef** *urlRef** *note** *noteRef** *sourceRef*?

Date and place reference are optional. Person role references refer to the event's role players. Event attributes hold other attributes of the event. Event references refer to other events this event may refer to as evidence. Using event references allows event records to be structured into trees of event records. These trees allow the full research process to be reflected in the database. See the section on Place records for the definition of *placeRef*. A *placeRef* can refer to a separate Place record or can contain the place information *in situ*.

eventType :: **type :** (**birth** | **death** | **marriage** | **baptism** | **burial** | **residence** | **immigration** | **naturalization** | ... | *userDefEventType*) **;**

There is a long list of available event types; I invent them as I need them.

personRoleRef :: **rolep : [** *id personRole personRoleRefAttr** **]**

Events refer to persons by person role references. Each contains the id of a person record, the role the person plays, and any attributes needed to clarify the role.

personRole :: **role :** *personRoleType* **;**

personRoleType :: (**father** | **mother** | **parent** | **child** | **husband** | **wife** | **spouse** | **brother** | **sister** | **sibling** | **uncle** | **aunt** | **cousin** | ... | *userDefPersonRoleType*)

There is actually long list of available person roles; this is the tip of the iceberg; I invent new ones when needed.

personRoleRefAttr :: *attribute*

An attribute of the role reference. Good examples include **age** or **occupation**, which serve to give the person an age or occupation at the time and place of the event.

eventAttr :: *attribute*

Any attribute required to document the event.

eventRef :: **event : [** *id eventRefAttr** **]**

Event references refer to other events that provide the evidence behind this event. Event records can therefore be structured into event trees using these references to represent growing hypotheses and conclusions about the events in the lives of persons in the database.

eventRefAttr :: *attribute*

Person Records

Person records hold information about people. Records may represent information extracted directly from evidence, or records may represent a researcher's hypotheses and conclusions about a real person.

personRecord :: **person : [** id *personContent* **]**

personContent :: *personName* gender personAttr* eventRoleRef? vital* relationRef* personRef* urlRef* note* noteRef* sourceRef**

The person's name is optional and has its own substructure. Event role references refer to events the person plays roles in. Vital substructures hold information about events the person is the primary role player in. Relation references refer to persons the person is related to. Person references refer to other persons this person refers to as evidence.

personName :: **name : [** *nameContent* **]**

Name content defines the substructure of person names.

gender :: **gender : (M | F | U) ;**

DeadEnds allows other strings, such as **male**, **female**, **unknown**.

personAttr :: *attribute*

eventRoleRef :: **role : [** *id eventRole eventRoleRefAttr** **]**

Persons refer to events by event role references. Each contains the id of an event record, the role the person plays in the event, and any attributes needed to clarify the role. Evidence persons refer to evidence events (and *vice versa*).

eventRole :: **type :** *eventRoleType* **;**

eventRoleType :: (**birth** | **death** | **burial** | ... | *userDefEventRole*)

vital :: **vital : [type :** *vitalType* **;** *vitalContent* **]**

A vital structure is event information placed in a person record rather than in a separate event record. This is useful in cases where the event has a primary role player, and the evidence only provides information about that person. The vital structure is placed in that person's record.

vitalType :: (**birth** | **death** | **baptism** | **burial** | ... | *userDefVitalType*)

vitalContent :: *date? placeRef? eventAttr* urlRef* note* noteRef* sourceRef**

Vital content is simpler than event content. It does not contain person role references.

relationRef :: **relation : [** *id relationRole* **;** *relationRefAttr** **]**

A relation reference refers directly to another person that this person is related to. Its **id** property is the id of the other person. The **relationRole** indicates the relationship this person has with respect to the other person. The other person may have a reciprocal relation back to this person. The reference establishes a relationship between the persons. The approach is useful if evidence establishes the relationship but gives no further information.

relationRole :: *personRole*

relationRefAttr:: *attribute*

personRef :: **person : [** *id personRefAttr** **]**

Person references refer to other persons that provide the evidence behind this person. Person records can therefore be structured into person trees using these references to represent growing hypotheses and conclusions about the person in the database.

personRefAttr :: *attribute*

Place Records

Place records represent locations where events have happened. Place records may be constructed into hierarchies. For example city place records may refer to the county they are in, which may refer to the state or province they are in and so forth.

placeRecord :: **place : [** *id placeContent* **]**

placeContent :: *placeType placeName placeAbbrev placeAttr* placeRef*?

placeType :: **type :** (**village** | **town** | **city** | **county** | **state** | **province** | **region** | **country** | ... | *userDefPlaceType*) **;**

placeName :: **name :** STRING **;**

placeAbbrev :: **abbrv :** STRING **;**

placeAttr :: *attribute*

placeRef :: **place :** (**[** *id placeRefAttr** **]** | *placeContent*)

Place references can refer to the record of the place that contains this place. This allows place records to be constructed into trees that represent sets of hierarchical places. The place model does not have provision for historical changes in locations. Third parties could provide sets of related place records. Instead of referring to another Place record a *placeRef* can also contain the content information of its parent place *in situ*.

placeRefAttr :: *attribute*

Note Records and Note Structures

General notes can be placed in separate records if many other records will refer to it. Otherwise notes can be included as node sub-trees within the records they apply to.

noteRecord :: **note : [** *id noteContent* **]**

noteContent :: **note :** STRING **;**

The model will be enhanced to allow styled text for notes.

noteRef :: **note :** (**[** *id noteRefAttr** **]** | *noteContent*)

There are both note records and note structures that can be properties of other records. Records are better if numerous records will refer to the same not. Internal properties are available for notes that pertain to only one record.

note :: **note : [** *noteAttr** **]**

Group Records

Group records represent an association of persons. The main use of this record is to form family records.

groupRecord :: **group : [** *id groupContent* **]**

groupContent :: *groupType date? placeRef? personRoleRef* groupAttr* urlRef* note* noteRef* sourceRef?*

groupType :: **type :** (**family** | ... | *userDefGroupType*) **;**

groupAttr :: *attribute*

Entity Records

Entity records represent institutions and other entities of interest in genealogical research. Examples of such entities include courts, cemeteries, hospitals, churches, colleges, banks, institutes, corporations and so on.

entityRecord :: **entity : [** *id entityContent]*

entityContent :: *entityType entityAttr* urlRef* note* noteRef* sourceRef?*

entityType :: **type :** (**court** | **cemetery** | **hospital** | **church** | **college** | **university** | **institution** | **association** | **corporation** | ... | *userDefEntityType*) **;**

entityAttr :: *attribute*

URL Records

URL records refer to information external to databases, often files on the local file system. Other records can refer to these URL records.

urlRecord :: **url : [** *id urlContent*]

urlContent :: *urlName urlType urlAttr* sourceRef?*

urlName :: **name :** STRING **;**

urlType :: **type :** (**file** | ... | *userDefUrlType*) **;**

urlAttr :: *attribute*

URL references are used in records to refer to URL records that point to information about the record.

urlRef :: **urlp : [** *id urlRefAttr** **]**

urlRefAttr :: *attribute*

Attributes

Attributes are used throughout the rules. Most rules in fact are special purpose attributes. In addition the user is free to add attributes. The rules indicate where these attributes can appear. For example in the *personContent* rule the *personAttr** component indicates where person attributes can appear. The *personAttr* rule is:

personAttr :: *attribute*

The variety of specific attribute rules are used to indicate that there are pre-established lists of attributes that are appropriate in each context, though the user may add attributes with

other tags. The attribute rule itself was not defined above, though the general discussion of nodes hinted at what it would be. Here is the rule defining attributes:

attribute :: *tag* **:** *content*

tag :: IDENTIFIER

content :: (STRING **;** | **[** *attribute** **]**)

An IDENTIFER is a sequence of Unicode characters making up a conventional identifier, and a STRING is a quoted string of Unicode characters.

Person Names and Dates

Person names and dates are both represented by node trees with specific structuring rules. Neither are defined in this document yet.