Importing Persons and Events to DeadEnds Programs from User-defined File Formats

The DeadEnds programs use an external file format for archiving genealogical data. The programs export and import data in this DeadEnds file format. In many cases, however, person or event data may already exist in other file formats, or may be easy to generate in other formats, and users may wish to import data in those formats.

The DeadEnds programs have an import feature that can read files holding information about persons or events. The files may be in many different text formats that are defined by header lines at the beginning of the files. The files are made up of *lines*, where each line is a sequence of *fields* separated by a special character.

Each external data file begins with one or more header lines that specify the format of the data lines that follow in the file. This allows the import function to understand the lines and fields, create the necessary DeadEnds Person and/or Event records, and put the field information into the right locations in the records.

For example, say we have extracted many death records from the social security death index, and we want to create DeadEnds Person records for each record. We must put the records into a file using a format the DeadEnds import function can read. We decide to use one line per record and break each line up into seven fields. Here is an example:

```
p : Thomas/Wetmore : M : 5 May 1896 : Nov 1970 : New London, Connecticut : 446-41-1729
```

The line is a fixed sequence of fields that holds a code, a person's name, sex, date of birth, date of death, place of last residence, and social security number. Each line requires a code value to help the import function recognize multiple line types. The colon is the default field separator character. (The white space on both sides of the colon are not required and ignored.)

In order for the import function to understand this information and build DeadEnds Person records from it, the file begins with the following single header line:

```
Person: "p" : name : sex : birth.date : death.date; reside.date : reside.place : ssn : source="SSDI"
```

This line is also a sequence of fields, but the information, instead of being actual data, describes the fields of the data lines. This is called meta-data because it describes data.

The first symbol in the header line, `Person`, specifies what kind of DeadEnds records are to be created from the data in the file. Currently Person and Event records are supported. The remaining fields correspond to fields in the data lines. The first, "`p`", specifies that each data line will start with a 'p' character. The remaining symbols are *path expressions* that define how each field in a data line fits into a DeadEnds record. Note the use of the semicolon in the sixth field, allowing the field to hold two

path expressions; this indicates that the data in this field is mapped to two locations in the Person records. Note that the last field defines a fixed value for the Person record's source. All Person records created will be given that value for its source value; therefore the individual data lines do not include source fields.

A DeadEnds Person record created from the data line above could be shown in GEDCOM format as:

```
0 INDI
1 NAME Thomas /Wetmore/
1 SEX M
1 BIRT
2 DATE 5 May 1896
1 DEAT
2 DATE Nov 1970
1 RESI
2 DATE Nov 1970
2 PLAC New London, Connecticut
1 SSN 446-41-1729
1 SOUR SSDI
```

Likewise the record could be shown in XML format as:

```
<person>
       <name> Thomas /Wetmore/ </name>
       <birth> <date> 5 May 1896 </date></birth>
       <death><date> Nov 1970 </date></birth>
       <reside>
              <date> Nov 1970 </date>
              <place> New London, Connecticut </place>
       </reside>
       <ssn> 446-41-1729 </ssn>
       <source> SSDI </source>
</person>
```

Defining Person Records That Span Multiple, Possibly Optional Lines

It is possible to define data files with more complexity. For example, the data for a person may be spread across different lines, and lines may be optional. Here is the header specification for a file that contains information about persons that may be spread across one, two or three lines.

```
Person: Attributes : BirthInfo? : DeathInfo?
Attributes: “p” : name : sex : occupation : religion
BirthInfo: “b” : birth.date : birth.place : birth.source
DeathInfo: “d” : death.date : death.place : death.source
```

The first line is mandatory and includes basic information about the person, here name, sex, occupation and religion. The second line is optional and holds birth information. The third line is also optional and holds death information. The fields in the first header line are not path expressions; they refer to symbols that must be defined in additional header lines. The question marks on BirthInfo and DeathInfo symbols indicate that either or both of the lines they specify are optional. An example of data lines that specify a person using this specification might be:

```
p: Thomas/Wetmore: M: Antiques dealer: Episcopalian
b: 5 May 1896: New London, Connecticut: New London vital records
```

In this case there is no line with death information.

Defining Record Clusters that Include an Event Record and One or More Person Records

The DeadEnds import function also supports events. This is a slightly more complicated case because DeadEnds represents each event as a related set of records, one Event record for the event itself, and one Person record for each of the persons who played a role in the event. Because of this, each file with event information must begin with a header line of the following form:

```
Event : EventPart : RolePart
```

The symbol `Event` specifies that the file contains event information. The next two symbols, `EventPart` and `RolePart` specify the formats of the data lines that hold the data about the event and the data about the persons. The names of these symbols can be any name the user chooses.

Here is an example of how the event and person lines might be defined for events taken from a census:

```
EventPart : “c” : type=”census” : date=”1851” : place : note : source=”1851 Census of New Brunswick, Canada”
RolePart : “r” : name : sex : age : role : occupation : religion
```

Here is an example of data using this format:

```
c : Norton, Kings County, New Brunswick : image 34, sheet 68 from Ancestry.com
r : Justus S /Wetmore : M : 56 :  Head : Farmer : Episcopalian
r : Deborah /Wetmore : F : 52 : Wife : : Episcopalian
r : Almira Jane /Wetmore: F : 13 : Daughter : : Episcopalian
```

After reading these four lines the import function would create one DeadEnds Event record and three DeadEnds Person records. Expressed using XML these records would be:

```
<event type=”census” id=”e1”>
       <date> 1851 </date>
       <place> Norton, Kings County, New Brunswick </place>
       <note> image 34, sheet 68 from Ancestry.com </note>
       <role type=”head” id=”p1”><age> 56 </age></role>
       <role type=”wife” id=”p2” ”><age> 52 </age></role>
       <role type=”daughter” id=”p3” ”><age> 13 </age></role>
       <source> 1851 Census of New Brunswick, Canada </source>
</event>
<person id=”p1”>
       <name> Justus S /Wetmore/ </name>
       <sex> M </sex>
       <occupation> Farmer </occupation>
       <religion> Episcopalian </religion>
       <role type=”head” id=”e1”/>
       <source> 1851 Census of New Brunswick, Canada </source>
```

```
</person>
<person id=”p2”>
       <name> Deborah /Wetmore/ </name>
       <sex> F </sex>
       <religion> Episcopalian </religion>
       <role type=”wife” id=”e1”/>
       <source> 1851 Census of New Brunswick, Canada </source>
</person>
<person id=”p3”>
       <name> Almira Jane /Wetmore/ </name>
       <sex> F </sex>
       <religion> Episcopalian </religion>
       <role type=”daughter” id=”e1”/>
       <source> 1851 Census of New Brunswick, Canada </source>
</person>
```

Note how the records interrelate. Also note that the ages are included in the Event's roles rather than in the Person records. The import function knows that ages are properties of roles in events rather than an intrinsic property of a person.

Path Expressions

Path expressions describe locations in hierarchical structures. They are composed of a sequence of names that represent the paths to locations in structures that begin at the structure's root. The best way to think of a path is first think of the record as a hierarchical tree of nodes. For example, a portion of the hierarchical structure of a DeadEnds Person record will have this tree structure:

```
person
       name
       sex
       birth
              date
              place
```

Path expression are then formed by using a dotted notation to show a path from the root of the tree to a leaf of the tree. In this example, all possible path expressions are `name`, `sex`, `birth.date`, and `birth.place`. Note that `person` is not included since it is always understood to be the root of the tree.