fhiso

CFPS 88
(Call for Papers Submission number 88)

Asynchronous Collaboration: A Proposal

Submitted by: Trauring, Philip

Created: 2013-07-30

URL: Most recent version: http://fhiso.org/files/cfp/cfps88.pdf
This version: http://fhiso.org/files/cfp/cfps88_v1-0.pdf

Description: A series of methods for sharing information between different researchers, that are investigating overlapping segments of their family trees, and receiving updates. The system allows users to collaborate, without having to be completely in sync. In addition, some ideas are presented on allowing the use of external databases for places, sources and events, and using UUIDs to connect different researcher researching the same people and events. This proposal is a method for exchanging data between researchers that: • Allows researchers to sync data between their trees without requiring the acceptance of all differences. • Allows researchers to receive updates on changes to collaborator trees, even if their trees are not completely in sync. • Enables the sharing of images and documents, even when those media files are very large (i.e. bigger than one could reasonably send via e-mail). • Creates a decentralized system of sharing, that is not dependent on researching using the same application or service. • Can facilitate finding other researchers that are researching the same individuals.

Keywords: collaboration, sharing, cloud storage, UUID, media, photographs, documents, sources, geographic, places, events

FHISO Call For Papers

Asynchronous Collaboration

aka AsyncGen

Submitted by: Philip Trauring
philip@trauring.com
http://lexigenealogy.com/

Abstract: A series of methods for sharing information between different researchers, that are investigating overlapping segments of their family trees, and receiving updates. The system allows users to collaborate, without having to be completely in sync. In addition, some ideas are presented on allowing the use of external databases for places, sources and events, and using UUIDs to connect different researcher researching the same people and events.

Keywords: collaboration, sharing, cloud storage, UUID, media, photographs, documents, sources, geographic, places, events

Created: July 30, 2013
Originally Published: http://lexigenealogy.com/2013/07/asynchronous-collaboration-a-proposal

Introduction

One of the major pain-points of researching one's genealogy is the process by which one shares information with relatives and other research collaborators. This is of course one of the main reasons there is a need for a modern standardized file format for sharing data. Another problem, however, is that when collaborating with other researchers, there is no easy way to track progress on common areas of interest. Today one is most likely to find out of another researcher's progress by receiving an e-mail from them, or seeing what they have posted on an online family tree. Why is there no way to exchange information directly when something changes in your overlapping tree?

This proposal is a method for exchanging data between researchers that:

- Allows researchers to sync data between their trees ***without*** requiring the acceptance of all differences.
- Allows researchers to receive updates on changes to collaborator trees, even if their trees are not completely in sync.
- Enables the sharing of images and documents, even when those media files are very large (i.e. bigger than one could reasonably send via e-mail).
- Creates a decentralized system of sharing, that is not dependent on researching using the same application or service.
- Can facilitate finding other researchers that are researching the same individuals.

Table of Contents

Choosing a Repository

The first step in collaborating between two researchers is to choose a common repository. A repository could use one of several technologies, based on open standards or proprietary technologies, and it could be free or subscription based. Examples of technologies that could be used as repositories are FTP, WebDAV, Dropbox, SugarSync, Amazon Cloud Drive, etc. A given genealogy application or service could support one or more of these services, or even provide their own service. As part of a standard, there should be at least one common repository technology required by all application/service developers.

Two researchers, for example, are using two different genealogy applications, but they both have Dropbox accounts. One creates a folder and shares it with the other, and they assign that folder as their repository.

Once a common repository is defined and configured in each researcher's application, each application would transfer a configuration file to the repository, defining it's capabilities, such as what formats and what versions of formats it can process. This will allow multiple file formats to be supported, and this feature to be implemented even before new file exchange formats are finalized.

It should be added that it is entirely possibly to eliminate the need for a remote repository, keep the data exchanged locally only, and exchange information directly over the Internet. Just like one can chat and exchange files over the Internet in real-time, one could exchange genealogy data the same way. In an ideal world, there would probably be a combination of approaches, allowing direct communication when available, and sending the data to the remote repository when it is not. The asynchronous nature of using a remote repository has many advantages, particularly if the researchers are in different time zones.

Getting in Sync

The next step is to determine which section of one's tree is to be worked on together. Usually this would include everyone in the tree who is related to both researchers (plus spouses), although it could be possible to set up a more restricted tree segment, or to define it completely manually (i.e. if the people are researching a family not related to one or either of them).

The simplest way to define the research area would be to choose yourself and your collaborator in your tree, and have the program itself determine all blood relatives and spouses and show the list to you to confirm. For future definition, the research area could be defined as descendants and ancestors of the most distant common ancestor.

Once the research areas are defined, the applications would create a tree file of the defined area, for comparison. This file could be GEDCOM or whatever subsequent standards come into existence. The designated interchange format is something that is

determined at the beginning, when each application views the configuration file of the other application, and determines the best format to use. Each app uploads its file to the repository, and then download the file from the other researcher. Loading the other researcher's file, the application then needs to lead the user through a match-and-merge process to get both databases in sync.

The sync need not be perfect – in other words both researchers need not have the exact same data in their databases. The reason for this is that this collaboration process is really meant to share changes, not necessarily accept changes. Thus, in this first stage, each user goes through the differences in their databases and accepts the changes they want. If they have questions about changes, they can use the query system described below. This is a major aspect of this proposal, that it enables people to share genealogical data without having to accept that data if they don't think it contributes to their tree (such as if there is no source, or the source is dubious).

Ongoing Changes

Once each researcher has gone through the differences in the initial tree file and accepted what they want, they now are ready to share ongoing changes. There are a couple of ways to handle this stage.

One way is to record all changes to the database in a log file (or a directory of log files), and later collect all the changes to people you are collaborating with, sharing the changes. Sharing the changes could be triggered based on several events – such as quitting your genealogy app, after a set amount of time (X minutes after the last change), manually choosing to share, etc.

Another method could be that every time one makes a change to their database, the application would check if the person was part of a research collaboration, and if so it would share the change. Multiple changes to the same person could be consolidated based on various events such as described in the above example. This is more or less the same system, but not tracking changes of people you are not collaborating on.

No matter how the changes are collected, a file is generated with all the changes made. This file could be all the changes to all the people your are collaborating with, or a series of files, one per person or other entity (i.e. the addition of an image to a source could be independent of an individual). The change file(s) would be uploaded to the repository. The next time the second researcher accessed their genealogy program, or if their program is already open it could check on a regular basis, the researcher would be notified there are changes and could step through them one by one and accept them if they choose.

If a researcher receives a change file indicating a change to a record they never accepted previously, or references a record they never accepted, the application can look back at the original sync file and determine the connection. For example, if the researcher never accepted a record showing a child of a known relative, and now a

record is sent showing a spouse of that child, referencing the child obviously, the application can still show you the connection and offer to add the child as well.

To make the above process easier, the application can keep a sync file updated at all times with all changes sent from the corresponding researcher. Thus at any time you can see exactly what your corresponding researcher's tree looks like and compare it with yours. This means that every time a change is received, regardless of whether you accept the change, this file would be updated with the changes.

Research Tasks

In addition to changes being shared, researchers collaborating could also share a list of tasks. Each researcher could post items they are looking for, and either researcher can complete the task. In some cases, the other researcher might already know the answer to the question being asked in the task, or they might just choose to do the research needed for the task when they have time. Tasks might be finding primary documents to back up information already known (or thought to be known), adding a photograph of the person, locating the person's gravestone, etc.

Although up to the application developer, it would be nice to have a kind of collaboration view that would show both a timeline of activities (i.e. showing each change done to your shared tree) as well as a list of pending tasks. A researcher could check out a task, so to speak, where they indicate they will be taking on the task, so the other researcher(s) don't work on it. Once the researcher completes the task, they can mark it complete by adding the information acquired. When the task is completed, the user's application could add an annotation to the original task file in the repository indicating it was completed.

The Query Process

When a user received a change file, they should be shown the data on the user in their records, and the new data being presented in the change file. They can choose to accept the change immediately, or if they are not convinced of the change, or otherwise has questions, they can send back a query to the researcher who created the change. For example, if the change consisted solely of adding a birth date, without providing a source, the researcher could ask the original researcher to provide the source for the data. When the original researcher receives the query, they should be allowed to immediately add a source citation to the data, whereupon it would be sent back to the second researcher, and they could then choose to accept it. What this means is that queries are structured to allow the applications to know what the query is, and not just a plaintext question. Plaintext queries could be allowed as well, but would be considered a different type of query.

A series of query types should be developed, with only the last option being a plaintext query. Some query types could include:

- What is the source for this change?
- Do you have other sources for this change?
- Do you have a document scan of the source mentioned?

Other queries could be sent even without having received a change from the other researcher. For example:

- Do you have a picture of this person?
- Do you have a source or sources for this piece of information?
- Do you have a scan of the document you cite as a source?
- Do you know the birth/marriage/death date for this person/couple?

Queries can be sent in the same way as changes, as a file or series of files, that are added to the repository, and processed by the other researcher's application. In response to a query, the receiving application can prompt the user to add whatever piece of information was requested, and then immediately share the new information with the other researcher. This will help researchers to add in the sources and documents they may have, but may have forgotten to add to their genealogy database.

Media Handling

One of the historical problems with sharing genealogy data has been the lack of media sharing. Media was never supported directly in the GEDCOM standard (or rather it was included, but never supported by applications). In the final draft of GEDCOM, 5.5.1 (never finalized, but still supported by many genealogy applications) released in 1999, direct media support was dropped in favor of links to media file locations.[1] The result of this lack of support is that sharing genealogy data almost never includes an automated way to bring in someone else's media files. This is another issue that can be solved by use of an online repository.

Media files, whether photos, videos, audio files, or scan of documents, etc. should be able to be associated with individuals, families, sources, events, places, etc. The specifics of these associations are not part of this proposal, it should just be mentioned that whatever associations are developed in future standards should also apply to sharing within this proposal. One particular feature that would be useful in this regard would be to be able to specify crop coordinates of a photo to associate with an individual. This would allow using one group photograph (of a family for example) to associate images cropped from that photo with each individual.

When a researcher shares a section of their tree with another researcher, the associated media files can also be shared. Depending on the repository (how much storage they allow, etc.) the researchers could exchange the original files, or smaller versions of them. For example, there could be a cap on image size enforced to keep

[1] https://devnet.familysearch.org/docs/gedcom/ged551.pdf

storage usage down, but the researcher could request a full-size image which would be uploaded to the repository, downloaded by the other researcher, and then deleted.

There could be three file sizes for images that are standard – thumbnail, preview and original. Applications could generate thumbnails of every image, as well as preview images which would be capped at a specific size such as 1000 pixels on a side and use higher compression levels, and that would allow people to see all the images, even without downloading all the originals which might take up more space than someone is willing to use.

Utilizing Universal Unique Identifiers

When implementing this system, there are a number of important uses for a Universal Unique ID (UUID) system. UUIDs, also sometimes called GUIDs (Global... by Microsoft), are very large (128 bit) numbers that can be generated by disparate systems without overlapping numbers (at least the probability of generating the same number is incredibly small).[2] This means that different people can create unique IDs for items, without having to compare them with a central registration server somewhere.

UUIDs are used in many genealogy programs, but are not standardized. FamilySearch proposed guidelines for 'GEDCOM Unique Identifiers' in 2000, which proposed using UIDs for all individual and family records, and keeping all UIDs associated with those records (i.e. if you import a GEDCOM with different UIDs for the same person, to keep all UIDs for that person).[3] Many programs use the _UID GEDCOM tag to export the UUID, although different programs use different formats for the UUID (some variations of the standard UUID, some not standard at all).[4]

The original version of the UUID standard was based on the MAC address of the computer it was created on, and the number of nanoseconds since the adoption of the Gregorian calendar.[5] This standard, called Version 1, has in many cases been discarded in favor of pseudo-random versions, but for reasons which largely go counter to the goals of the genealogist. The reasons newer versions of the UUID algorithm were developed was specifically because the original version was linked both to the computer that they were created on (through the MAC address) and the time it was created, which created privacy concerns in many applications. However, in a genealogy sharing framework, these features are actually beneficial, and the added benefit that the UUIDs are actually guaranteed to be unique instead of just statistically likely to be unique, is helpful to keeping the matching process simpler. This means the preferred version of UUID for genealogy purposes, and thus for the purpose of this proposal, is Version 1.

---

[2] http://tools.ietf.org/html/rfc4122

[3] https://devnet.familysearch.org/docs/gedcom/GEDCOMUniqueIdentifiers.pdf

[4] http://www.tamurajones.net/The_UIDTag.xhtml

[5] http://en.wikipedia.org/wiki/Universally_unique_identifier#Version_1_.28MAC_address.29

With large web-based services that may have thousands of users running on the same computer, a separate unique identifier could be used in place of the MAC address. This would insure that records were still unique between users of their service.

In this system, UUIDs could be used for more than just identifying individuals in records. Each researcher could have their own UUID, to link to their research. This researcher UUID (RUUID) could have several purposes. For one, each time a change is shared with other researchers, the RUUID would be listed in the file, insuring that the change is coming from a recognized research collaborator. In addition, the RUUID can be connected to original sources or images, so that the origin of those records is known. For example, if one researcher shares their personal photos, their RUUID could be associated with those images (embedded using image metadata standards like IPTC, EXIF, XMP, etc. or even embedded directly into the images using steganography techniques).

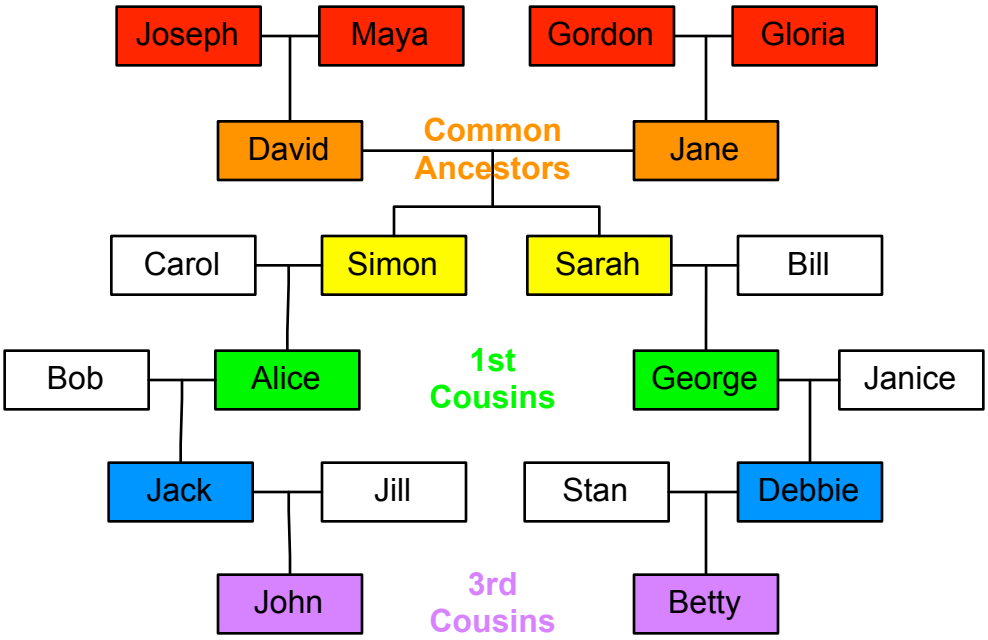
Other Uses

Another use of the processes described in this proposal, would be to keep two separate applications that you are using in sync. For example, if you are using a person-centric genealogy application and a separate source-centric genealogy application, you could in theory use these methods to keep the two applications in sync, even if they contain different information for each user. You could also use this approach to sync your genealogy database with an application or service that is multimedia-oriented, such as a book publishing service or family slideshow application. If in the time you are working on an extended project the data changes, you could use this to update the service(s).

A Use Case – John and Betty

This is a free-flowing example. There is no intention for the XML documents outlined here to be examples of the standard being proposed – that should be derived from existing or future standards. The examples shown here are simply for illustration.

John and Betty are third cousins. They share common great-great-grandparents. They found each other on the Internet when researching their families. John had posted a family tree on a major online genealogy web site, and Betty found his tree online. John had built a tree that went up to his gg-grandparents, David and Jane. He knew his great-grandfather Simon had a sister Sarah, but didn't know who she had married or any of her descendants. Betty, as a descendant of Sarah, knew her side of the tree. She also knew the names of her ggg-grandparents, which John did not. Sarah found John's tree because she was searching for her gg-grandparents David and Jane on a big genealogy web site, and came across David and Jane in John's published tree. John used the web site he had published his family tree to to also build and manage his tree. When he found new relatives he added them to this online tree. Betty had only recently started her genealogy search, but had decided on using software on her computer.



A look at the family tree of 3rd cousins John and Betty

Betty asks John if they can share their data. John agrees, but how? Both John's web service, and Betty's computer application, support online repositories for exchanging data. They check which services each one supports, and decide to use Dropbox. One of them sets up a folder and shares it with the other. They then set up a collaboration in their respective applications, and designate the Dropbox folder as the repository. Once that is done, the following files get uploaded to the repository folder:

John-RepoConfig.XML

```
<CONFIG>
  <HEAD>
    <RESEARCHER>
      <UUID>c8690c00-
e700-11e2-91e2-0800200c9a66</UUID>
      <NAME>John</NAME>
      <EMAIL>john@john.com</EMAIL>
      <SHARE>PUBLIC</SHARE>
    </RESEARCHER>
    <DATE>2013-05-29 10:23:45</DATE>
    <APP>AncestryFamilyOnline</APP>
  </HEAD>
  <FORMAT>
    <SYNC v="5.5.1">GEDCOM<//SYNC>
    <SYNC v="1.0">STEMMA</SYNC>
    <SYNC v="2.0; 3.0">GedXML</SYNC>
    <CHANGE v="1.1">AsyncGen</CHANGE>
    <IMAGE>JPEG</IMAGE>
    <IMAGE>TIFF</IMAGE>
  </FORMAT>
</CONFIG>
```

Betty-RepoConfig.XML

```
<CONFIG>
  <HEAD>
    <RESEARCHER>
      <UUID>db1e4c7a-e700-11e2-9d96-
f23c91aec05e</UUID>
      <NAME>Betty</NAME>
      <EMAIL>betty@betty.com</EMAIL>
      <SHARE>LINKED</SHARE>
    </RESEARCHER>
    <DATE>2013-05-30 12:11:34</DATE>
    <APP v="10.03">Reunion</APP>
  </HEAD>
  <FORMAT>
    <SYNC v="5.5.1">GEDCOM</SYNC>
    <CHANGE v="1.1; 1.2">AsyncGen</
CHANGE>
    <IMAGE>JPEG</IMAGE>
  </FORMAT>
</CONFIG>
```

Each application uploads their respective configuration file to the specified repository folder. John's online service indicates in his config file that it supports GEDCOM 5.5.1, STEMMA 1.0, or GedXML 2.0 or 3.0 for the initial sync, version 1.1 of the AsyncGen change file standard (something I just made up here), and JPEG and TIFF image files. Betty's application only supports GEDCOM 5.5.1 for the initial sync, can use version 1.1 or 1.2 of the AsyncGen change file standard, and only supports JPEG image files. Each application now knows that the only common formats they can handle are GEDCOM 5.5.1 for initial sync, AsyncGen 1.1 for change files, and JPEG for image files. In a final system, it would probably not be worthwhile to support multiple sync formats but rather to create a sync file format which would probably just be whatever new format is created through the FHISO process. The change file would also be derived from this format as well. The reason I specify multiple formats is if genealogy companies want to add support for their own proprietary formats here, they should be allowed to do so, but there should be a common format required as part of the standard to insure that all applications can interoperate. In addition, there may be a requirement for a minimum of support for one or more image formats, such as JPEG, but that shouldn't prevent application developers from adding support for more formats (such as TIFF, PNG, etc.). Another reason to allow multiple formats is to allow the adoption of this process before a final format is completed. For example, as most genealogy programs have a way to match-and-merge GEDCOM files, a GEDCOM file could be used as an initial sync file until a new format emerges.

When the config files are uploaded and each researcher's application has read the corresponding files, they determine the limitations of their communications – i.e. which file formats they each understand and can exchange.

Another thing to note is the **<SHARE>** tag, which specifies the extent to which the researcher is willing to share their contact information with other researchers. I envision the options as Public, Linked and Private. Public means you're willing to share your contact information even on public trees on the Internet, Linked means that you only agree to share your contact information with other researchers who have collaborated with people you've collaborated with and that are researching the same people, and Private means you are only providing your contact information to the person you are collaborating with and they should not share it with anyone. This is part of the external tree matching, discussed later in this document.

John and Betty then each select the portion of their trees they want to share with each other. This can be done automatically by asking the applications to share all common relatives (plus spouses), or by selecting the most recent common ancestors (i.e. David and Jane). Through either method, the application could determine a list of relatives to share. In addition, by selecting the section to share based on relationships, the applications can continue to analyze additions to each tree, and see if the new individuals fit within the definition specified. Let's say John and Betty both have trees that cover their great-grandparents on down through their own branches. Their applications would then upload the initial sync file to the shared repository. They might look something like:

John-2-Betty-InitialSync.xml

```
<TREE>
  <HEAD>
    <RESEARCHER>
      <UUID>db1e4c7a-e700-11e2-9d96-f23c91aec05e</UUID>
      <NAME>John</NAME>
      <EMAIL>john@john.com</EMAIL>
      <SHARE>PUBLIC</SHARE>
    </RESEARCHER>
    <DATE>2013-05-29 10:23:45</DATE>
    <APP>AncestryFamilyOnline</APP>
  </HEAD>
  <FAMILY id="4">
    <UUID>346d30a8-ea25-11e2-9064-f23c91aec05e</UUID>
    <PERSON id="10">
      <UUID>346d34e0-ea25-11e2-9064-f23c91aec05e</UUID>
      <NAME>
        <DISPLAY>David Baran</DISPLAY>
        <GIVEN>David</GIVEN>
        <GIVEN alt="birth" lang="Polish">Dawid</GIVEN>
        <GIVEN alt="dim" lang="Polish">Dawidek</GIVEN>
        <SURNAME>Baran</SURNAME>
      </NAME>
      <GENDER>MALE</GENDER>
      <EVENTS>
        <BIRTH>
          <DATE>1888-MAY-01</DATE>
          <PLACE authority="geonames.org" id="769591" permalink="http://
www.geonames.org/769591/kanczuga.html">Kańczuga, Poland</PLACE>
          <SOURCE id="34">
             <UUID>346d3904-ea25-11e2-9064-f23c91aec05e</UUID>
          </SOURCE>
        </BIRTH>
        <DEATH>
        <DATE>1950-DEC-12</DATE>
        <PLACE authority="geonames.org" id="4930956" permalink="http://
www.geonames.org/4930956/boston.html">Boston, MA</PLACE>
        </DEATH>
      </EVENTS>
    </PERSON>
  </FAMILY>
  <FAMILY id="1">
    <UUID>346d3c60-ea25-11e2-9064-f23c91aec05e</UUID>
    <PERSON id="1">
      <UUID>346d3fb2-ea25-11e2-9064-f23c91aec05e</UUID>
      <NAME>
        <DISPLAY>John Smith</DISPLAY>
        <GIVEN>John</GIVEN>
        <SURNAME>Smith</SURNAME>
      </NAME>
      <GENDER>MALE</GENDER>
      <EVENTS>
        <BIRTH>
          <DATE>1988-JAN-04</DATE>
          <PLACE authority="geonames.org" id="4930956" permalink="http://
www.geonames.org/4930956/boston.html">Boston, MA</PLACE>
        </BIRTH>
      </EVENTS>
    </PERSON>
  </FAMILY>
</TREE>
```

Betty-2-John-InitialSync.xml

```
<TREE>
  <HEAD>
    <RESEARCHER>
      <UUID>c8690c00-e700-11e2-91e2-0800200c9a66</UUID>
      <NAME>Betty</NAME>
      <EMAIL>betty@betty.com</EMAIL>
      <SHARE>LINKED</SHARE>
    </RESEARCHER>
    <DATE>2013-MAY-30 23:45:01</DATE>
    <APP v="10.03">Reunion</APP>
  </HEAD>
  <FAMILY id="7">
    <UUID>ab8c5e40-ec53-11e2-91e2-0800200c9a66</UUID>
    <PERSON id="5">
      <UUID>ab8c5e41-ec53-11e2-91e2-0800200c9a66</UUID>
      <NAME>
        <DISPLAY>Jane Baran</DISPLAY>
        <GIVEN>Jane</GIVEN>
        <GIVEN alt="nickname">Janee</GIVEN>
        <SURNAME alt="birth">Brown</SURNAME>
        <SURNAME alt="married">Baran</SURNAME>
      </NAME>
      <GENDER>FEMALE</GENDER>
      <EVENTS>
        <BIRTH>
          <DATE>1890-APR-21</DATE>
          <PLACE authority="geonames.org" id="4943629" permalink="http://
www.geonames.org/4943629/medford.html">Medford, MA</PLACE>
        </BIRTH>
        <DEATH>
        <DATE>1956-JUN-23</DATE>
        <PLACE authority="geonames.org" id="4930956" permalink="http://
www.geonames.org/4930956/boston.html">Boston, MA</PLACE>
        </DEATH>
      </EVENTS>
    </PERSON>
  </FAMILY>
  <FAMILY id="1">
    <UUID>ab8c5e42-ec53-11e2-91e2-0800200c9a66</UUID>
    <PERSON id="1">
      <UUID>ab8c5e43-ec53-11e2-91e2-0800200c9a66</UUID>
      <NAME>
        <DISPLAY>Betty Baker</DISPLAY>
        <GIVEN>Betty</GIVEN>
        <SURNAME>Baker</SURNAME>
      </NAME>
      <GENDER>FEMALE</GENDER>
      <EVENTS>
        <BIRTH>
          <DATE>1985-FEB-05</DATE>
          <PLACE authority="geonames.org" id="5380748" permalink="http://
www.geonames.org/5380748/palo-alto.html">Palo Alto, CA</PLACE>
        </BIRTH>
      </EVENTS>
    </PERSON>
  </FAMILY>
</TREE>
```

Keep in mind this is not using any kind of standard. It's just meant to be readable pseudo-XML to illustrate what is going on. If some of the ideas here are useful in the discussion for a new standard file format, that's great, but it's not the primary focus of this proposal. What I've created is a simple XML structure that includes a header (**<HEAD>**) with information on the file, families and persons (**<FAMILY>** and **<PERSON>**), places (**<PLACE>**), sources (**<SOURCE>**), and media files (**<MEDIA>**). As I want to keep code to a single page, I will be presenting the code piece by piece.

Above we start with the header and the tree itself. I've only added two people per tree as this is just an example. In John's tree I've shown John and Betty's gg-grandfather David, and John's own record. In Betty's tree I've similarly shown John and Betty's gg-grandmother Jane, and Betty's own record.

Like in some genealogy programs, I've assigned UUIDs to families and persons. I've also assigned UUIDs to sources and media. For places, I reference an external geographic database (in this case geonames.org). There need not be a reliance on a single place name database, which is why the geographical database authority is listed. Below, I'll show more about Places.

We start with the header (**<HEAD>**). Inside the header we identify the researcher (**<RESEARCHER>**) by name (**<NAME>**) and by UUID (**<UUID>**). Presumably we could include all kinds of other information if we want here - physical address, social media information, etc. but some basic information is all we need. This could be up to the researcher to determine.

The IDs shown within other tags (**<FAMILY>**, **<PERSON>**, **<SOURCE>**, etc.) are not something to necessarily be imported into the other person's tree, although they can be used to delineate relationships and thus must be imported in some fashion (but not necessarily as actual ID numbers). They're really intended to make the document more readable. A genealogy program usually has internal ID numbers for specific entities in the tree, and IDs in this file can (and should) map to the exporting application's own internal IDs. This allows one to review the sync file and compare it to data in your genealogy application in case you run into problems. When reviewing the sync file it would also be difficult to compare UUIDs, and a shorter ID number is helpful. In additional, once someone starts using this system, they are likely to have multiple UUIDs for each person and family, thus making it that much more difficult to make sense of the file without a single shorter number to use to identify specific families, persons, etc.

The reason a file is likely to have multiple UUIDs (at least after the initial sync) is that collecting all the UUIDs for an individual allows for better syncing over the long run, and among different users. If you sync with one researcher, and then sync with a second one who has already synced with the first one, then you will already have the UUIDs from the first sync, and will be able to merge data much faster. You can also use the

UUIDs to find other researchers on external trees. I discuss external tree matching later in this document.

John and Betty each open their respective genealogy application (John's is web-based, Betty's is on her computer) and see the initial sync file has been downloaded and processed. Each goes through their application's match-and-merge process to update their trees with the information present in the other researcher's file. Whenever a person or family is matched to a person or family in the other researcher's file, the UUID from that file is added to the local record. If this is the first collaboration for each of them, then for each person they share in their file, there should be two UUIDs. If they import people that they don't have themselves in their tree, then they should just import the UUID for each new person, and there would only be one.

As Betty examines the new information, she notices that in the record for David, their common gg-grandfather, John has listed a birthdate, whereas she only knew of an estimated birth year. She sees the source listed as a Polish birth certificate, with all the details, but wants to know if John has a copy of the certificate itself. She thus initiates a query, which sends a file to the repository something along these lines:

```
<QUERIES>
    <HEAD>
    ...
    </HEAD>
       <QUERY type="mediarequest" id="1" >
              <SOURCE id="34">
                     <UUID>346d3904-ea25-11e2-9064-f23c91aec05e</UUID>
              </SOURCE>
              <COMMENT>Wow, you have our gg-grandfather's birth certificate? I'd
love to see a copy. Can you attach an image of the certificate? Thanks, Cousin
Betty.
              </COMMENT>
       </QUERY>
</QUERIES>
```

The header would be the same as sent in other files, showing Betty's contact info, privacy settings, a timestamp, etc. I've left that out to save space. The query has four components, only three of which are required. The first is the query type, which is a 'mediarequest'. This tells the application that the researcher is requesting a media file (i.e. an image) of the item.

The second is the id number. This is just a sequential series of numbers of the queries between the two researchers. Each time a query is generated, a number is assigned to it for tracking purposes. When a response is sent it will contain the same id number.

The next is the item itself, the source which is referenced by it's UUID. In this case I've also added the local ID, which matches John's application's ID, since that was shown in the original sync file. This isn't necessary, but is still useful for troubleshooting purposes. How to determine whose id this is would be important to have in the specification. The third item, which would be optional, would be a free-text comment. The query itself is

structured so the other researcher will be asked to add the media item if they have it, but adding a comment personalizes the request.

John would receive the query and his application would show him the request. If John has the image, he could respond by immediately adding an image of the birth certificate. This would add the image to his own tree, as well as send a copy to Betty via the repository. Let's assume John has the image. He would add the image, and the following file would be sent to the repository:

```
<QUERIES>
    <HEAD>
    ...
    </HEAD>
    <RESPONSE type="mediarequest" id="1" response="1">
              <SOURCE id="34">
                     <UUID>346d3904-ea25-11e2-9064-f23c91aec05e</UUID>
              </SOURCE>
              <MEDIA>
                     <UUID>346d4318-ea25-11e2-9064-f23c91aec05e</UUID>
                     <CREATOR>
                            <NAME>John</NAME>
                            <UUID>346d3fb2-ea25-11e2-9064-f23c91aec05e</UUID>
                     </CREATOR>
                     <DESC>Birth Certificate for David Baran, May 1, 1888</DESC>
                     <FILE>davidbaran-birthcert.jpg</FILE>
              </MEDIA>
    </RESPONSE>
</QUERIES>
```

Along with this file, the media file itself, *davidbaran-birthcert.jpg*, would also be uploaded to the repository, and placed in a directory for images. This could be permanently stored, or set up to be deleted after it is downloaded by the other researcher (or even after a set period of time).

If the original file was not a JPEG, but something else such a TIFF, the application would recognize that Betty's application can only import JPEGs, and automatically convert the image to a JPEG first.

Let's take a look at the response. First it mirrors the original query with the same type, and the same reference to the original source. The id is 1, the same as the id for the query. The response adds `response="1"` which indicates simply that the response is positive. In this example, if John didn't have the image, it would be set to `response="0"`, and the response would only include the reference to the source, without any additional information. Optionally, there could be a comment in the response as well, whether it was positive or negative.

In this case, it's a positive response, so there's also a reference to the media file itself. Included with the image information is the UUID assigned to the image by John's application, as well as John's name and researcher UUID. This information would be optional, but if John is the person who found the birth certificate and scanned it, this

would identify that fact. The information on John could also be inserted directly into the image via EXIF or similar meta-data techniques.

Next are an optional (but preferred) text description of the media file, and the name of the file itself. As part of the standard, where this file would be located would be pre-determined, but it could also be specified here. For example, media files could be organized by creation date, or by import date, or by type, or whatever criteria is considered the best method.

Not shown above, but possible, would also be to add information directly about the person the document refers to - i.e. a person tag that links to David and includes his UUID. In reference to the query this isn't strictly necessary since that information is already known, but it might be useful for troubleshooting and making the document easier to read.

It should also be possible with queries such as this to sent the information without actually adding it to one's own tree. For example, if John had the image but did not want to post it on his public tree, he could send it directly to Betty, but not include it on his tree.

Now John and Betty are essentially in sync. This is not that different from where we stand now when people share a GEDCOM. The query system is an improvement in that it automates getting answers for some of the questions that arise from syncing trees, but we're still stuck at the same point more or less where people are when they share a GEDCOM today.

Like many researchers who collaborate with their distant cousins, maybe nothing happens for a long time. Perhaps months go by, and finally one of them decides to do some more research into that specific family line. Maybe they received an e-mail from an online genealogy site that introduced a new databases that piqued their interest. Let's say John received the e-mail and did a search which unearthed a new piece of information – the birth location of John and Betty's ggg-grandmother Gloria. Betty had shared the birth date of Gloria with John, but she didn't know where Gloria had been born. In the new database John searched, the birth location is revealed. John adds the birth location to his family tree, and cites the online database he searched. A change file is generated, and uploaded to the repository. It might look something like:

```
<CHANGES>
	<HEAD>
	...
	</HEAD>
	<CHANGE id="1">
		<PERSON id="11">
			<UUID>ab8c5e44-ec53-11e2-91e2-0800200c9a66</UUID>
			<DISPLAY>Gloria Brown<DISPLAY>
			<BIRTH>
				<PLACE change="add" authority="geonames.org" id=""
permalink="http://www.geonames.org/5140538/tarrytown.html">Tarrytown, NY</PLACE>
				<SOURCE change="add" id="65">
					<UUID>346d4660-ea25-11e2-9064-f23c91aec05e</UUID>
					...
				</SOURCE>
			</BIRTH>
		</PERSON>
	</CHANGE>
</CHANGES>
```

Standard header. The **<UUID>** tag is used to identify which person is being modified. There is no real need to put the name, but for troubleshooting purposes I've included the display name. One could argue to add the other information such as given and surname tags, and the birthdate (names repeat in families), to make things that much clearer. For processing purposes none of them are needed, but if a human being wants to look at the file, knowing who is being referred to is useful.

A **<CHANGE>** tag has an id number which is a sequential ID number assigned to each change request for tracking purposes. Probably this should not be called *id* but something more unique, to keep it distinct from the other IDs in use in the file. It's debatable whether these ids should be separate or all from the same sequence. For example, there are ids assigned to changes, queries, research tasks, etc. and each could have their own sequence of numbers, or these could all be pulled from a single sequence. I think it probably makes sense to keep them separate for each type, and track each item in an index file that lists all the changes, queries, research tasks, etc that have been generated in this collaboration. Perhaps cid, qid, rid, etc.

Inside the **<CHANGE>** tag is as much of the outside tags as needed to get to the tag being changed – so there is a **<PERSON>** tag and a **<BIRTH>** tag because those are needed to get to the **<PLACE>** tag which is being changed. Indicating the change is a simple `change="add"` which indicates that the change being performed is the adding of new information. Options for this setting beyond *add* could be *remove* and *replace*, and possibly others for unique situations.

Everything inside the tag marked *add* should be added. The second change is the source, which is also new. In this instance, the source data is included directly in the listing, which is different then how it was done in the initial sync file (where it was referenced). In this case I think this makes more sense since it is being added, and is a direct reference of the birth record. Note that the **<UUID>** tag and everything else inside the source listing need not have a `change="add"` added to them, since everything

inside a changed tag should be added/changed. I've left out the details of the source, but it would be referencing the online database where the information was found.

I Want My Places

One big problem with keeping genealogy files consistent has been the wide variety of ways that people write out place names. Keeping things consistent between multiple researchers can be difficult, especially when different people want to see different things. One of the big advancements in recent years has been the inclusion of geographic databases to genealogy programs, that allow one to link to a specific location from the database. In some cases, genealogy software publishers have generated their own place name databases for use with their software. The problem with this solution sometimes is how to display the location. If I choose Boston, MA as a location, a database might show:

> Boston, Suffolk County, Massachusetts, United States

for a town in Poland, I might get:

> Kańczuga, Kańczuga, Powiat Przeworski, Podkarpackie, Poland

The reason Kańczuga is listed twice is that it's the name of the town and the local administrative district. This is true in the United States as well where some counties share the name of their largest city. In any case, the point is that while all of that information is useful to have, it's not something I want to display whenever I show the town name. I might want, for example, *Boston, MA* and *Kańczuga, Poland*. Another researcher from outside the US might prefer that the country is added to US listings, and display *Boston, MA, USA*, while a researcher in Poland might prefer a listing that drops the country in favor of the district like *Kańczuga, Podkarpackie*.

There are a couple of ways to fix the display issue. One is to allow the user to choose the specific administration levels to show for each location. This would probably need to be set on a country-by-country basis. In the above example, for the US I would choose to show the town name and the state name (skipping the county and country), and for Poland I would display the town name and the country (skipping all three administrative districts in between).

You might be thinking this is dangerous because there may be more than one town in one of the intermediary districts with the same name. Indeed in the case of Kańczuga there are two towns with the same name in the same administrative district. The reason this is not a problem is that the location is still linked to the actual location in the geographic database, and depending on the application you are using, you can always find the extended information on the location.

If you look in the initial sync file for John above, you'll notice the following line in the birth record for his gg-grandfather David:

```
<PLACE authority="geonames.org" id="769591" permalink="http://www.geonames.org/769591/kanczuga.html">Kańczuga, Poland</PLACE>
```

This specifies the birth location as Kańczuga, Poland, and links to the location in an external geographic database. Any number of geographic databases could be used for this purpose, although they should be public databases to insure anyone can check them. The text of the place name here is how John wanted to display it. It could similarly specify to show the town and country for this place name. What is good about this format is that if the importing researcher doesn't like the format John has used, they can leverage the full database and choose what information to show (i.e. force it to conform to the format they've chosen for place names from that country).

To make things easier, and to insure that one can look back and understand what a researcher intended, I suggest including a place name database in any file exchanged (including just the place names mentioned in the file). For example, in the above example where the simple place information was linked to a birth record, later in the sync document, there would be a **<PLACES>** section that could look like the following:

```
<PLACES>
  <PLACE authority="geonames.org" id="769591" permalink="http://www.geonames.org/769591/kanczuga.html">
    <NAME>Kańczuga</NAME>
    <DISTRICT>
      <COUNTRY>Poland</COUNTRY>
      <ADMIN level="1" id="858788">Podkarpackie</ADMIN>
      <ADMIN level="2" id="7530898">Powiat Przeworski</ADMIN>
      <ADMIN level="3" id="7533486">Kańczuga</ADMIN>
    </DISTRICT>
    <LOCATION>
      <LAT>49.98346</LAT>
      <LON>22.41168</LON>
    </LOCATION>
  </PLACE>
</PLACES>
```

This internal database would just mimic the external database and insure that a researcher looking at place names in another researcher's file would know exactly which place the researcher was referring to when they list a place. In other words, for every place reference in the file being exchanged, a full place record would be added to the **<PLACES>** section of the file. If for example the third-party database the person used ceases to exist, a researcher looking at the original file will still have all the important information about the place. This is also a good reason to insist that all geographic databases used are public. This would allow occasional snapshots of existing geographic databases, to insure that if a database for whatever reason closes up, that the reference numbers used in peoples files can be translated to another geographic database. Other reasons it's important to have the database public, is that one can also build source templates on top of a geographic database, and you wouldn't want to lose access to the geographic database you're basing your source templates on (see Sources below).

Geonames.org is a very good public place database, and although it does not yet fully support historic place names, it is something they are considering. As we develop genealogy standards, we should be helping efforts like geonames.org to fully support the needs of genealogists in their place name databases.

Sources

There are two issues with citing sources in genealogy files – how they are input and how they are output. Output is a question of display. One can prefer Chicago citations, or MLA citations, or EE citations, but in the end it is just a display issue. The most important issue when creating a citation is that all proper information is collected, and that it is saved in such a way that it can but output in a readable way. The trick is of course to standardize on what each piece of information is called. Whether you display the author name of a book citation first or second in the display doesn't really matter compared with recording the author's name in a way that it can be recalled as the author's name. Where this becomes complicated in particular is when you try to include sources from many different countries. Some countries use the term Fond to refer to a collection of documents – should a Fond number be labeled as Fond, as it is described in that country, or a Collection, the word in English, or perhaps both? These issues are something that need to be worked out for successful recording of sources such that they can be used by future researchers.

In addition, just like geographic data can be organized into public databases, it is also possible to organize source templates around public databases. While geographic databases have more uses than genealogy, it should still be possible to build public databases of source templates, built on top of a geographic hierarchy. For example, let's say we already have a geographic database that contains the following sample hierarchy:

Jurisdiction	**Name**
Country	United States
State	Massachusetts
County	Suffolk County
Municipality	Boston

We have a 4-level hierarchy that generally corresponds to how records would be organized as well. For example, some records would be at the country level (federal records), some at the state level (i.e. state court records), the county level (i.e. property deeds and probate court records) and the municipal level (i.e. vital records).

For each record type, there are a certain number of fields which are needed to insure that the same record can be found again quickly. These fields different for different

record types, but also for the same record types in different locations. It's possible, even likely, that the fields for a specific type of record in the same location might differ during different time periods. This is particularly true in countries that went through many changes in rulers, such as Poland, where the same record types in the same locations but different times can be found in Polish, Russian or German depending on when they were recorded. One might also want to record the fields in their original language, and in English, or French, or whatever language you prefer. If a database of source templates is built, there can be a master language (the original) set, and then volunteers can offer translations for different languages. This is another disconnect between function and display, which is important.

A sample source record, continuing the example above:

```
<SOURCES>
  <SOURCE id="34">
    <UUID>346d3904-ea25-11e2-9064-f23c91aec05e</UUID>
    <CITER uuid="db1e4c7a-e700-11e2-9d96-f23c91aec05e">John</CITER>
    <CITATION>
         <TYPE>BIRTH</TYPE>
         <LOCATION>
              <PLACE authority="geonames.org" id="761168" permalink="http://
www.geonames.org/761168/przemysl.html">Przemyśl Archive</PLACE>
              <URL>http://www.przemysl.ap.gov.pl/</URL>
              <FOND>1731</FOND>
              <YEAR>1888<YEAR>
              <AKTA>94</AKTA>
              <SYGNATURA>523</SYGNATURA>
         </LOCATION>
         <DETAILS>
              <DATE>1888-MAY-01</DATE>
              <GIVEN>David</GIVEN>
              <SURNAME>Baran</SURNAME>
              <FATHER>
                <GIVEN>Joseph</GIVEN>
                <SURNAME>Baran</SURNAME>
                <TOWN authority="geonames.org" id="769591" permalink="http://
www.geonames.org/769591/kanczuga.html">Kańczuga, Poland</TOWN>
              </FATHER>
              <MOTHER>
                <GIVEN>Maya</GIVEN>
                <SURNAME>Koswolski</SURNAME>
                <TOWN authority="geonames.org" id="769591" permalink="http://
www.geonames.org/769591/kanczuga.html">Kańczuga, Poland</TOWN>
              </MOTHER>
         </DETAILS>
    </CITATION>
  </SOURCE>
</SOURCES>
```

Let's take a look at a few things here. First, the source has a local ID number, which is from the exporting application. This is for readability and for easy trouble-shooting. The source also has a UUID making it unique. There is also a UUID identifying the researcher who contributed the source. These UUIDs will allow someone to go through their records and see who contributed the source citations for specific pieces of information. The record also gives the specific location of the source, in this case in an

archive in Przemyśl, Poland. Everything needed to locate the specific record in the archive is listed.

External Tree Matching

One of the interesting benefits of using this system is that one can build a decentralized database of researchers who are related to, or otherwise researching, the same people. As people connect to other researchers, they share their contact information with them. Researchers should be able to specify how widely they want their contact information shared. In the example above, John allowed public access to his contact information, and Betty only allowed linked researchers to see her information.

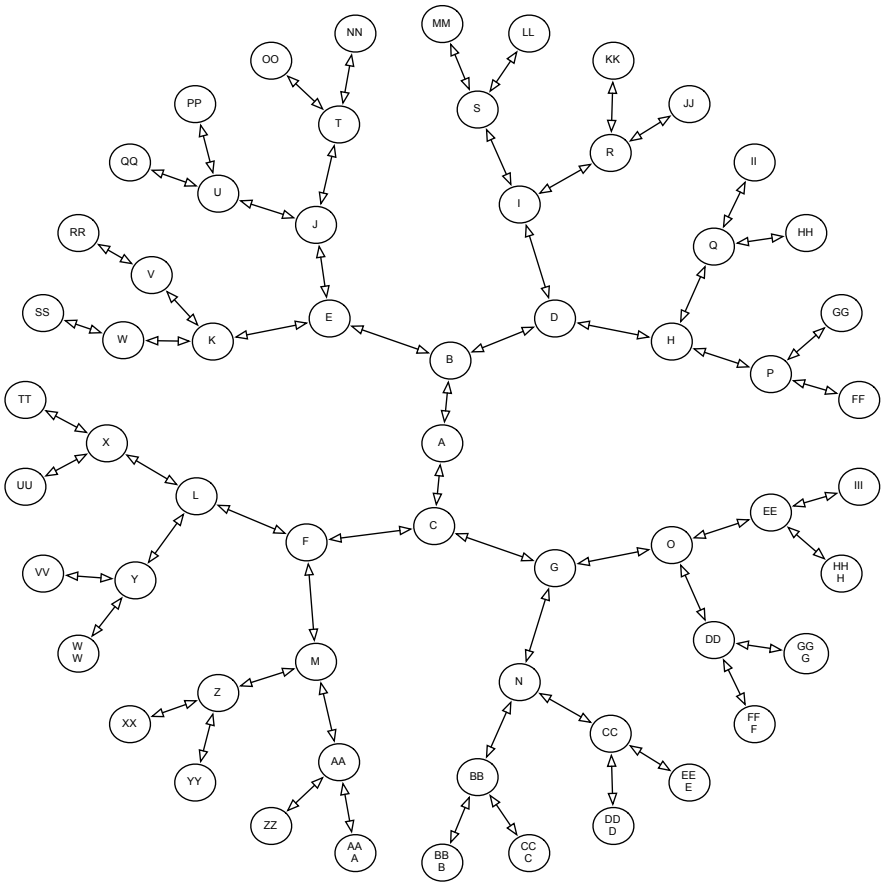
Privacy Levels for Sharing Contact Information	
Public	Anyone can see
Repo	Anyone in same repository can see
Linked	Only people linked to researcher through other researchers can see (can be capped at # of generations)
Linked Repo	Only people linked to researcher through other researchers in the same repository can see (can be capped at # of generations)
Private	Only the people a researcher shares with can see

Let's look at how this could be implemented. One option is that the repository being used could be purposely built for genealogy data exchange. An existing genealogy service or application developer could build a free or subscription-based site to use as a repository. One of the features they could offer as a benefit over using a generic site like Dropbox, is leveraging the contact information and settings of files on their site. For example, if you add your files to a repository, and the repository finds other researchers with people who have the same UUIDs in their files, or even matches people based on specific information (i.e. matching names, dates, etc.) the repository site could contact each researcher and let them know there are other researchers on the site researching the same people.

In addition, different repository sites could theoretically exchange some information to help find matches. One repository could send a list of UUIDs to another through a special API, and get back a response if there are matching records with those UUIDs. Alternatively, one or more central servers could be set specifically for matching purposes, and many repositories could connect to those servers and check for matching UUIDs across all repositories.

To illustrate the potential in connecting to other researchers in this approach, look at the following diagram which shows each researcher adding two connections. You start with

one researcher, who adds two, making three. Those two new researchers add two each, making seven total. The numbers rise exponentially, and that assume no other additions by the original researchers.



Connection growth where each person adds two connections

Within 5 steps in this diagram, you've reached 60 other researchers. Realistically most people won't find this many people who are researching large sections of your tree, but you can find that many people researching parts of your tree. Some may only overlap a single person in your tree (such as if they're related to a spouse of someone in your family) but that may be exactly the connection you would never otherwise find. That spouse's family might have no idea what happen to their great-grandfather's sister Mabel who moved far away when married, you can fill in her married life, and they might have a photo of her that you don't.

Additionally, if two researchers are connected through a match, they can be automatically connected via a repository to exchange information. This will depend largely on whether they have accounts on the same repository service, etc. but even if they use totally different applications, totally different repositories, and even speak different languages, the connection can still be made.

Another interesting byproduct of this feature, is that the most recent change file uploaded by a specific researcher can be used to determine how recently a specific researcher has been active. This date can help determine if the contact information may be potentially out of date, or if the date is sufficiently old – if perhaps the researcher may be deceased. A researcher could even determine rules on their research availability based on their usage. For example, a researcher could say that if they don't access the repository for two years, then automatically make the most recent version of their tree available to researchers researching the same people. This could be a way to insure that the work they've done is not lost.

Events

Another area where the expanded use of UUIDs can benefit genealogy is in defining events. There can be external databases used here as well, to include major historical events, but where things really could get interesting is finding connection between people based on the events they attended. For example, if an event is a wedding and has a UUID, it can be attached to the people in the wedding (who also have UUIDs) i.e. the bride and groom. Now whenever that event or those people are merged into another researcher's file, the UUIDs are combined.

Repositories that allow external searching could show not only the people that match a search, but the events associated with those people. Imagine in the wedding mentioned above that you have a photo album from the wedding. You've scanned all the photographs and tagged everyone you know in the photos. Each photo is thus associated with the people in the photo, and with the event itself. The event is associated with all the media files, and all the people in all the photos. Even if someone is not in your tree, but in a photo from the wedding of someone in the tree, they could be made searchable online.

Besides enabling people you may not even know or have had connections to in generations to find photographs of their family members, eventually this could be used to do more advanced analysis such as finding people who show up at many of the same events as people in your family, but is not someone in your tree. This could lead to people to contact to ask about your family, people who may have photographs that include your family members, etc.

Commercial Deployment

There are several ways that genealogy software companies can take advantage of the ideas in this proposal. It is certainly my hope that genealogy software companies would implement these ideas in their own applications, particularly in a way that is standards-based and interoperable. There are other ways companies and organizations can benefit from and contribute to the use of the methods described in this proposal. I wanted to include a few ideas.

Software companies and especially web-based application providers, can deploy repositories. These repositories can be part of whatever services they currently provide, or be an add-on service. These services can also be tiered to allow for different amounts of storage, etc. Companies that provide web-based family trees can provide everything behind the scenes between their own members, but can also provide access to uses of different applications and services.

Genealogical societies can also create their own repositories for their members. These can allow members to use the features within a much more controlled environment, and members might be more willing to share their contact information with other members of a society repository, as opposed to a large public repository.

Once a full specification is developed, it would be important to include a certain basic set of features that would be required to claim compatibility. This should include file formats and methods for exchanging data. This might include the new FHISO exchange file format, JPEGs for images, and WebDAV for file exchange. Support for one or more external databases should also be considered, such as geographic databases. Commercial companies should be free to add support for other formats and databases, including proprietary formats. As long as a basic set of formats is supported, everyone will be able to work together. This allows commercial companies to both support repositories, but also differentiate their products from each other.

Conclusion

It's my hope that this proposal will spark a conversation on how people can collaborate with genealogy research more effectively. Along with modernizing the data formats used to exchange genealogy data, we need to also introduce new methods for exchanging and updating data, and standardize those methods among all the stakeholders.

Philip Trauring
July 30, 2013