An API to create Shared List in TCIA/NBIA

Background

At it's core, NBIA consists of two parts: the user facing 'web' application and the backend database. Shared lists are created in the 'web' application as means for storing pointers to a set of SeriesInstanceUIDs that can be shared and reused to access a subset of data. A shared list cannot be made at the patient/study level. Shared lists are stored in two tables in the NBIA database. These tables are *custom_series_list* and *custom_series_list_attribute*. The former stores metadata about the shared list (Name, Who created it, When etc.), while the latter contains a list of SeriesInstanceUIDs. that are contained in a shared list. There is a one-many mapping between the two tables. Figure 1., provide a description of these two tables.

CUSTOM_SERIES_LIST —1————————N—> CUSTOM_SERIES_LIST_ATTRIBUTE

custom_series_list_pk_id	Primary Key of this table
Name	Name of shared list
Custom_series_list_time stamp	Date (not time) of creation
Hyperlink	URL
Comment	Text description
Username	TCIA username of creator

custom_series_list_attribute_pk_id	Primary Key of this table
SERIES_INSTANCE_UID	The SeriesInstanceUID that is being saved
custom_series_list_pk_id	Primary key of corresponding custom_series_list

Figure 1: The schema of the two tables that manage Shared Lists in TCIA/NBIA. There is a one-many relationship between custom_series_list and custom_series_list_attribute and *custom_series_list_pk_id* in each row of the CUSTOM_SERIES_LIST_ATTRIBUTE table maps to an entry in the CUSTOM_SERIES_LIST table.

How Shared Lists are created in NBIA

When a user creates a shared list via the 'web' application, a new entry is made in the custom_series_list table. User provides the name of the shared list, some comments and a URL. The application determines the date and the username. Finally, the NBIA application (and not the database) assigns primary key (*custom_series_list_pk_id*). Next, it creates a set of entries in the CUSTOM_SERIES_LIST_ATTRIBUTE table by inserting the SeriesInstanceUIDs, and the primary key that was created for the custom_series_list table. Again the NBIA application (and not the database) assigns primary keys for each entry in the custom_series_list_attribute table. These primary keys are created by a Hibernate HiLo strategy.

Challenges with creating Shared Lists via a API

An API that attempts to create a shared list will need to provide the metadata and the contents of the shared list via a HTTP POST. There is one major and one minor challenge when attempting to create shared lists via the API.

The minor challenge comes from the fact that there is no way to programmatically resolve an API-Key to a username. This can be easily dealt with by changing the input parameters of the create api and requiring a user to provide their username.

The major challenge relates to the primary key. When attempting to create a shared list via an API, one would move the burden of key generation to the database. This would be done by altering the table so that the primary key column (*custom_series_list_pk_id*) AUTO INCREMENTS

```
ALTER TABLE ncia.custom_series_list modify column CUSTOM_SERIES_LIST_PK_ID
bigint(20) auto_increment;
```

By doing so, the API can easily add data to the two tables. However, we now run into the issue where there are two different interfaces to adding data (NBIA and the API) and consequently two different strategies (HiLo by NBIA and the database by API) that are being used to generate the primary key (*custom_series_list_pk_id*). This can lead to a situations where NBIA attempts to insert a record and generates a primary key that already exists in a record that was inserted by the API. Such key-clashes can lead to unpredictable behavior by NBIA and subsequently impact TCIA user experience where a user tries to save a shared list and that list is not recorded or it appears to have been recorded but is never saved to the database.

To test this, I used the TCIA-Dev instance (https://public-dev.cancerimagingarchive.net/ncia/login.jsf). I changed the shared list metadata table so that the primary key autoincrements. Next, I added a create shared list API to the Bindaas dev instance. Finally I ran a sequence of create operations that alternated between creating a shared list via the API and the NBIA web application: API —> NBIA Web Application —> API —> NBIA Web Application.

The inserts from the API worked without a glitch. This was expected, as the generation of primary keys is handled by the database. However, when using the NBIA Web Application, the second insert failed. A subsequent retry would occasionally work. This unpredictable behavior is what led me to delay the “create shared list API”.

In addition to the issues with primary key, I also have some security concerns. In our current design, we require that all users who use the API must do so with an API-Key. However users are unlikely to remember their API-Keys. Therefore there is a risk that in applications that leverage this API, one user could inadvertently use another users API-Key and by doing so create shared lists on their behalf. This problem can be solved by deploying STS 2.0 in front of the TCIA LDAP, so that applications can authenticate their users using their TCIA credentials, and the various TCIA APIs, can guarantee the identity of their consumers.

Alternative Strategies

We develop a system that uses MongoDB to store shared lists that are created by an API. To distinguish the two, let's call these replica sets: Shared Lists —> TCIA & Replica Sets —> API.

With a replica set in place, users will have an API to create a replica set and consume replica sets. They will also have an API that allows them to consume shared lists. The only thing one won't be able to do is go to TCIA and browse replica sets. This shortcoming may be a minor issue since the intended users of APIs are developers who wish to integrate their apps with TCIA. Our driving use cases came from Andriy Fedorov and Jayashree. They wanted to be able to create, share, and consume subsets of data from within 3DSlicer and HubZero — almost like a Dropbox where pointers to data are shared, but the data stays on TCIA.

SharedLists store seriesInstanceUIDs. A replica set could be at a patient/study/series level. Therefore creating a replica set could be much more efficient, as one could reference an entire patient with their PatientID, rather than having to first retrieve all seriesInstanceUIDs and then create a shared list. There is an additional advantage with replica sets. SharedLists can only reference TCIA seriesInstanceUIDs and any binary blobs that we associate as Annotations. With replica sets we would be able to associate TCIA and collocated non-TCIA datasets.