XNAT Desktop Technical Specification

Quick Index

1. General

The proposed system (XNAT Desktop) will represent rich client software installed locally by users of one or more XNAT Enterprise deployments. XNAT Desktop is a **file-centric** architecture for managing local research files based on tagging system and interacting with XNAT Enterprise deployments. XNAT Desktop should efficiently deliver the following tasks:

- Track and manage files/folders specified by a user both locally and remotely;
- Attach additional metadata to tracked files/folders;
- Provide an automated way for the user to label individual files/folders;
- File search by name or label;
- Transmit labeled files/folders and metadata between local client and server XNAT deployment.

2. General software architecture and implementation notes

XNAT Desktop (XND) will consist of two parts: *XND Frontend* and *XNAT File Server*.
XNAT File Server will provide core functionality for managing repository item records (this is performed by repository manager) and transferring items between different File Server entities (performed by file transfer manager). *XND Frontend* will provide GUI to
a) manage item tags in local or remote repository,
b) query for items with specific tags, both locally and remotely,
c) transfer items between local and remote repositories.

3. Definitions

Repository.	A warehouse containing all files/folders tracked by XNAT Lite.
Repository root.	A root folder for managed items. Repository can have an arbitrary (but presumably small) number of roots for managed items. Each repository root should have a unique alias understood by XNAT file server (i.e. binding exists between system-addressable URI pointing to actual location on a local file system, and root alias). All references to repository items are maintained using root alias instead of actual file URI alias. This helps reduce security risks by hiding local file system structure and also add flexibility to managing
Container.	Each folder maintained under repository root, is referred to as a repository container. An actual path maintained by Repository is relative to one of existing roots. For instance, if we have a root “C:\documents and settings\testuser\my documents\data” with alias “/mri_data”, its subfolders “ /mpr”, “/recon” and “/seg” are maintained as “/mri_data/mpr”, “/mri_data/recon” and “/mri_data /seg”. Note that except for the repository root, none of the containers can be addressed by name different from the name on actual file system. Additionally, similarly to Amazon’s “digital bucket” concept, the relation between two different containers is not explicitly defined (although it can be inferred from the folder path).
Item.	File sitting in one of the folders addressable as repository containers. An expression “an item belongs to this container” implies that the actual file is residing in folder denoted by the container. An item that belongs to a container may or may not be managed by Repository database, and search queries to File server cannot identify an item in container unless it is managed.
Store / de-store item.	Add/remove item to/from Repository (start/stop managing item). Note that this operation does not involve file creation, copying or deletion in any way, but rather managing records in the database.
Basic item operations, file server operations.	Rename, delete, move, copy item. This functionality is performed by a file server which is conceptually separated from repository manager, due to architecture considerations, as discussed in “Architecture” section.

Tag.	Item tag is a “name-value” pair. Tag name is uniquely represented by alphanumeric string without spaces, defined by the user or predefined in XNAT Lite. For each item, each specific tag is either defined or undefined, and if defined, tag value can be an arbitrary string.
To tag item.	To define specified tag for selected item(s) in Repository.
Tag update.	Following the tag rename or delete, correctly update tag definitions of all in repository.
Software roles.	Each function described henceforward, will be logically ascribed to either XNAT Desktop or XNAT File Server, so this will be implied in expressions like “exists in Desktop role”, “included in File Server role”.

3.1 Basic Features

3.1.1 Tag Management

Zero, one or more tags can be attached to each item in repository.
The user should be able to:

1. Define a new tag;
2. Modify or remove existing tag (followed by tag update)

3.1.2 Item tracking

The user should be able to:

1. Browse through local file system, perform basic item operations and store selected items;
2. Browse through local/remote repository using pre-defined tag relationships, perform the same basic item operations as in a), de-store and tag/untag selected items;

3.1.3 Item search

XND will provide simple item search by combination of any number of tags and partial item name.

3.1.4 Interaction with XNAT Enterprise

XND should provide both client and server functionality for transferring repository items between XNAT File Server installations. For each upload session/item, minimal data the user will have to provide:

- Authentication information;
- Subject ID;
- Session ID.
- ??

3.2 Advanced Features (optional or to be implemented at a later stage)

Advanced features may include:

- Storing additional separate metadata with each item, or defining a metadata object based on one or more standard XNAT Enterprise element types – (UPD: this is supposed to be done using tag combinations?)
- Providing mechanism for sharing items in repository with other users on network (UPD: through the web-service interaction between different XNAT File Server installations)
- Consistent way of downloading items from XNAT Enterprise, and keeping track of Enterprise-specific unique identifier information (SessionID, SubjectID);
- Synchronization mechanism between local (file-centric) and Enterprise (session-centric) user repositories;

3.3 Networking: sharing repository resources

Currently, the need also exists to provide a generic API for sharing items in Repository over the network. The proposed architecture is divided, for implementation purposes, into DB query module (Repository manager) and resource (i.e. mainly file) transfer module (i.e. core File Server manager)

3.4 User interface

The user interface should provide the following:

1. A main frame graphical window with a list of main commands available to the user. Main commands can be available from menus (docked and/or pop-up) and/or toolbar(s), depending on capabilities of underlying technology. Main commands can be global or context-specific (“Connect to XNAT Enterprise” - global command, “Apply label to selected items” – context-specific).
2. Multiple-panel browsing window(s) for both repository and local file system. Repository items can either be displayed in a separate window (perhaps combined with a search window), or color-coded to indicate repository/non-repository items.
3. Window for displaying items in repository search results.
4. Pop-up dialog for managing labels (add, rename, remove);
5. Pop-up dialog for specifying item upload parameters;
6. Dialog for editing/specifying serializable configuration options.

Related: **XNAT File Server API**