XNAT Desktop Manual

Browse the source code: https://bitbucket.org/mmilch01/xnd

Download XNAT Desktop: https://bintray.com/nrgxnat/generic/xnat-desktop

Quick Index

Make sure to also check out this visual tutorial for tagging and uploading your data to XNAT server on Slicer3 wiki: http://www.slicer.org/slicerWiki/index.php/Slicer3:XND
If you have a problem with running XND you can ask question in XNAT discussion group.

1. General

XNAT Desktop (XND for short) application is purposed for organizing local files of arbitrary content, storing text metadata with managed files, distributed sharing of structured research data and measurements among desktop users, file repository servers and XNAT Enterprise archives. Each record in XND database contains two major elements: first, a resource location (i.e. file/collection path) on local or remote system, and second, a set of name-value pairs (referred here as tags). XND is highly extensible in regards to a specific domain of application, so there are no intrinsically built-in hard coded proprietary tags that cannot be modified; moreover, xnd can be used to create customized context hierarchies which specifically target local needs.

2. System requirements and installation

Prerequisites. The software is developed in Java 1.5 using Eclipse Rich Client platform v.3.3, and is tested to work on Windows 2000+, Linux with GTK, and MacOS 10.4.x+. 1 Gb RAM is required and 2+ Gb RAM is recommended to work with projects containing 20,000 files or more.
XNAT Enterprise compatibility. Upload to XNAT Enterprise currently works with XNAT.

Installation. To install, download the latest version for your operating system from https://bintray.com/nrgxnat/generic/xnat-desktop, and unzip.

> ATTENTION MAC AND LINUX USERS: make sure to check that xnd executable has "execute" permission before trying to run it. Refer to XNAT discussion group for detail.

Launch using "xnd" executable in the installation folder.

3. Terminology

- **Repository**. - Database containing records of all resources (files and collections) tracked by XND.
- **Repository root.** - A root folder for managed resources. Repository can have several roots. Each root should have a unique folder name. A root cannot be a subdirectory of another root. All references to resource in repository start with the root name.
- **Local resource.** - Managed file or collection in one of the folders under repository root. Resource has corresponding record (is managed) in Repository database; unmanaged files are not affected by any of XND operations and are not considered resources.
- **Collection.** - A set of files in one directory treated as a single resource for efficiency reasons. Collections are created with collection generation rules. For local repository, you can list collection members by double-click; files belonging to collection are shown as
- **Remote resource.** - File/collection managed by remote repository.
- **Manage/unmanage**. - Add/remove resource to/from Repository database. This operation does not involve file creation, copying or deletion in any way, but only operations on records in Repository database.
- **Tag (resource tag)**. - Name-value pair. Tag name is uniquely represented by alphanumeric string, defined by the user or predefined in XND. There is limited support for tags with multiple values.

- **Ontology.** - XND ontology is a collection of default tag definitions, categorical relations between tags (e.g. "Project is a parent of Subject"), and, optionally, pre-defined values for default tags (these are convenient to define for tags that assume finite subset of values, e.g. tag 'Modality' with values from the subset of (MR, CT, PET, CR, US, ...)).

- **Unmanaged files.** - Unmanaged files are shown as 🗋, you can 'manage' them, but all other operations (tagging, uploading, etc.) are not applied to unmanaged files even if they are part of selection.

- **Virtual directory (virtual folder).** - Virtual directory shown in VDS view represent hierarchical tag context of your data. For instance, virtual directory "Project:pr1->Subject:subj1" shows all managed resources which have tags Project="pr1" and Subject="subj1".

4. Basic tag operations

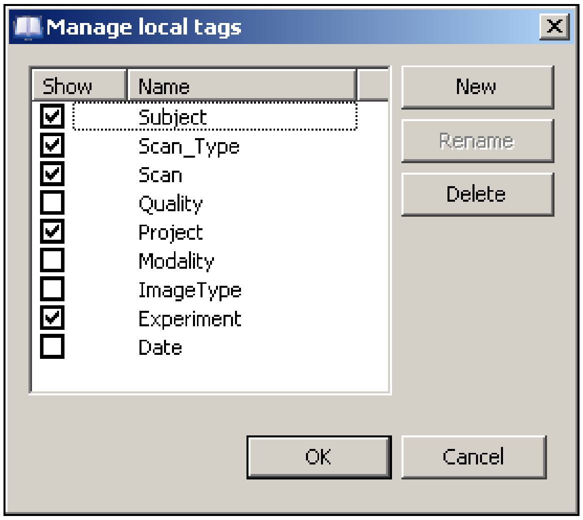
Define repository root(s). First, you'll need to point XND to one or more directories under which your managed files are located. Use "Add Managed Directory" toolbar command (or "Repository->Add dir" menu command) to select a directory. You can select multiple roots. To remove a root, use "Delete" context menu command on it. NOTE: this will not delete the folder on the file system.

> HINT: for best performance and convenience, choose the lowest level directory in the file system under which your data is located. For several unrelated datasets/project, it can be convenient to use multiple repository roots.

Add items to repository. Select folder under repository root and use "Manage" context menu command to add all files in the subtree to repository. The inverse operation is "Unmanage". All operations in context menu can be used recursively at any level on all applicable types of single items or multiple selections both in tree and table view.

> NOTE. Using context menu command on container object (such as folder or virtual folder) will propagate the command effect on all managed children of this container.

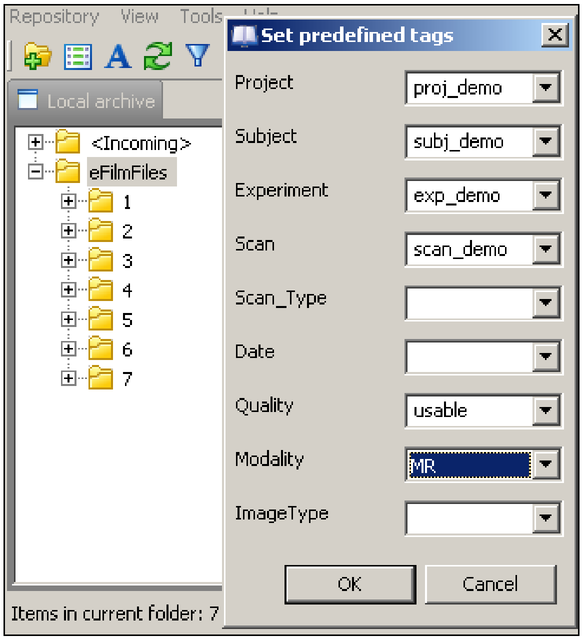
Define tags. Open tag management dialog by "Manage tags" toolbar command, or Repository->Manage tags menu command. (You'll see some tags already defined by default. These are XNAT ontology tags used by XND in communication with XNAT Enterprise installations. If you wish to re-distribute XND with your own set of tags, refer to Defining Custom Ontology). You can define any additional arbitrary tag names. Enable "show" checkbox to display column with values of this tag in the file view table.



Show	Name
☑	Subject
☑	Scan_Type
☑	Scan
☐	Quality
☑	Project
☐	Modality
☐	ImageType
☑	Experiment
☐	Date

Attach tag values to files. Use "Set tag" context command to specify tag value. Some tags (such as modality) can have pre-defined values that you can choose from the drop-down list. You can assign values for all or some pre-defined tags (those listed in default ontology xml) in one step, using "Set default tags" command.

> NOTE. Click on individual file to see all tags associated with it in the panel below the file table.

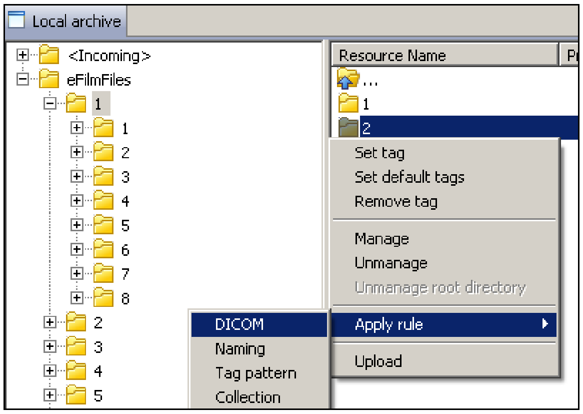


Set predefined tags

Project	proj_demo
Subject	subj_demo
Experiment	exp_demo
Scan	scan_demo
Scan_Type	
Date	
Quality	usable
Modality	MR
ImageType	

5. Automatic tag assignment

In many cases, underlying data files already contain meta fields that can be automatically translated to the useful "tag-value" pairs in XND. This is done through mechanism of tag generation rules. Each rule creates tag values for a given file based on contents (such as DICOM rule), previously existing tags (tag pattern rule), or context in directory structure (naming rule). To generate tags with specific rule, use corresponding rule command from "Apply rule" context submenu. As an example, DICOM rule will parse all valid DICOM files in selection, and generate XNAT-specific tags based on DICOM tags. You can see which DICOM tags are used, and also define your own DICOM rule by modifying rule definition XML file (see Defining custom DICOM rule).



6. File collections

The concept of file collections was introduced to improve performance of xnd when it manages large (20000+ files) datasets. File collections are simply lists of files which are treated by xnd as a single resource for purposes of tagging, that is, each file in collection shares the same tags, and all operations in xnd are performed on collection rather than individual files of that collection. Currently, collections can be generated in two ways:

1. Collection rule. When this rule is applied on selection of items, these items are sorted into collections based on "Project-Subject-Experiment-Scan" tag equivalence. That is, two files are assigned to the same collection if and only if values for all aforementioned tags are present and are exactly equal. So, you can tag your data with these tags and then apply Collection rule on them to convert records for each individual file into record of one collection containing all files with these tags. Note that as a result of such operation, tags that do not share the same value among all files that are assigned to the same collection, may not be preserved correctly. For instance, if two images in DICOM series have a tag "Date" with values "20090101" and "20090201", only the first value will be preserved in collection.
2. DICOM rule. DICOM series can be naturally combined into collections as there is rarely a need to tag individual image files, but rather a series as a whole. XND can generate collection from

DICOM series as DICOM rule is progressing. You can toggle collection generation in DICOM rule by changing the checkbox View->Preferences->Ontology and Rules->Generate collections when using DICOM rule.

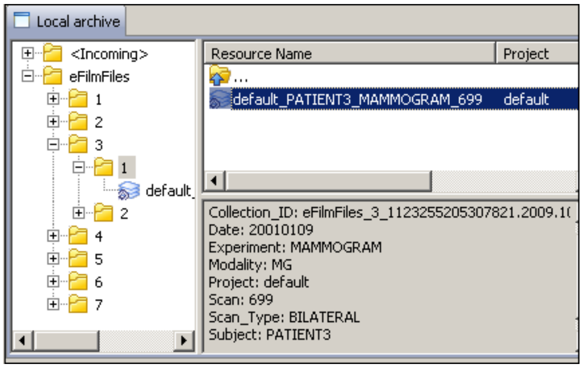
7. Using advanced data views

Hierarchical tag view. Initially, your data is shown in the folder tree/ file view, similar to any popular file browser. You can also view your data in Virtual Directory Structure (VDS) defined by tag hierarchy. XND comes with pre-defined XNAT VDS (XVDS) based on XNAT-specific tags, but you can always customize it or make your own (see Defining custom ontologies for details).

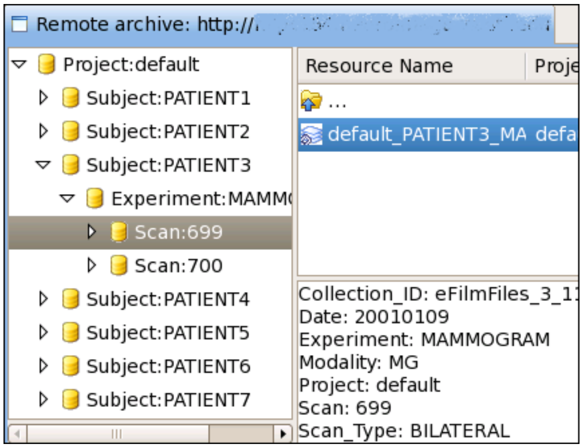
First, make sure you defined at least Project (and, optionally, Subject, Experiment, and Scan) tag for all files you want to display in XVDS. The DICOM rule will generate all of these tags automatically from DICOM files.

Next, use "Switch between folder and tag view" A toolbar or "View->Folder/tag view" menu command to switch to the VDS view. Most context menu operations are also supported for this view, with some obvious exceptions.

> NOTE. If you don't see expected hierarchy after switching to VDS view, or some files are missing from the view, that could mean that a) missing files don't have a root tag defined (for instance, that could happen when 'Project' tag is undefined for any of your files, and 'Project' is the only root defined in xnat_tags_default.xml); b) there is a problem with your custom ontology xml document.



Collection_ID: eFilmFiles_3_1123255205307821.2009.1
Date: 20010109
Experiment: MAMMOGRAM
Modality: MG
Project: default
Scan: 699
Scan_Type: BILATERAL
Subject: PATIENT3



Collection_ID: eFilmFiles_3_11
Date: 20010109
Experiment: MAMMOGRAM
Modality: MG
Project: default
Scan: 699
Scan_Type: BILATERAL

> NOTE. If you modify (add, remove, change) value of one of the tags that are involved in hierarchical display (i.e. Project, Subject, Session, etc. for the default hierarchy), use "Refresh view" toolbar command to see the changes in VDS.

Console view. XND has a console output, where progress for lengthy batch operations (e.g. networked file transfer) is reported. Console view is visible by default and can be re-opened with View->Console view menu command and is refreshed automatically when visible.

Remote repository view. XND supports single remote view of another XND repository instance, using XNAT File Repository web services API. Current version does not support user authentication, so use caution. To make your repository visible to another instance of the XND application, setup server connection in "View>Preferences" dialog. To view this repository from another computer, configure network address and port of the remote machine hosting XND in the View->Preferences->Client connection section, and use "Open remote view" toolbar or "View->Remote view" menu command. In case you need to reconnect, use "Connect to remote repository" command. An example of remote repository browsing is shown on Figure 4.

> NOTE 1. Only VDS view of remote repository is allowed, so if remote tag hierarchy is not defined or does not match the local tag hierarchy (for instance, it was modified as described in Defining custom ontologies), you may not see anything even if there are items in remote repository.

> NOTE 2. You can arrange various views (local, remote and console) within the application window frame using drag-and-drop on the view title.

8. Exchanging Files Between xnd Installations.

xnd supports uploading and downloading resources with metadata to/from remote repository. Use "download" and "upload" context menu commands on selected element to transfer files from/to remote repository. You can upload/download multiple selections of folders, virtual directories, or individual files /collections (see NOTE 4).

> NOTE 1. File transfer messages are shown in console.

> NOTE 2. Files uploaded from remote repository (i.e. when transfer was initiated by remote XND installation), are stored under special "<Incoming>/uploaded" folder. <Incoming> folder is always shown in managed roots. Downloaded files are stored by default under <Incoming>/downloaded folder, or you can set xnd to open file selection dialog to select location for downloaded files, by checking "View->Preferences->File Transfer->Select download location manually" checkbox.

> NOTE 3. You can specify the <Incoming> directory location through the main menu, View -> Preferences -> File transfer -> Incoming folder. By default, it is set to <your user directory>/.xnd/Incoming.

> NOTE 4. In current version, you can upload collections (they will be converted to individual records), but collection download is not supported.

9. Uploading data to XNAT

XND comes with pre-defined tag hierarchy (ontology) mapped to XNAT Enterprise data model, which makes it possible to tag your data for uploading to XNAT host directly from XND.

XND supports upload of XNAT MR sessions (roughly corresponding to Study level in DICOM terminology) on per-session basis. In XNAT ontology, MR session is an example of experiment (hence the 'Experiment' tag). To upload an experiment (including non-DICOM images and arbitrary resource files) to XNAT server:

1. 'Project', 'Subject' and 'Experiment' tags are required to be defined for all files. You can set some of them manually or using one of automatic tag assignment rules, depending on your data. If you defined subjects through XNAT website, make sure your local subject tags match subject labels in XNAT. You can also let XND to create subjects when uploading data, and edit them later, to enter subject-level demographic information supported by XNAT but not XND.

2. Also, set 'Modality' tag to 'MR' for all files you wish to upload (only MR session upload is currently supported by XND). If you used DICOM rule for tagging MR DICOM data, it's already done. Additionally, define 'Scan' value for raw images. All other files will be uploaded as XNAT session resources.

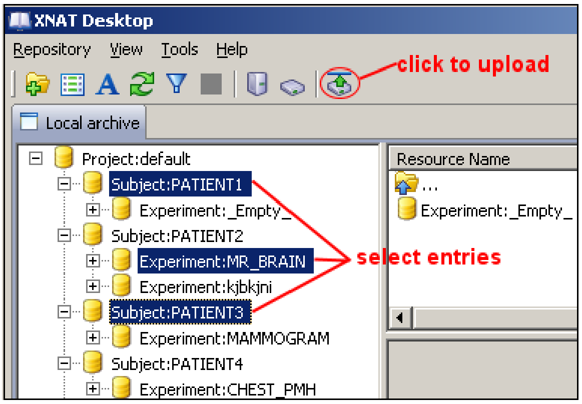
3. (Optional). You may define custom tags for session resources, they will be shown under 'resource tags' for each file (see below).

XND tag	XNAT field	Remarks
Project	Project abbreviation	In XNAT, check at project main page -> Edit Project Details -> Project Abbreviation
Subject	Subject ID	Must be defined on XNAT host before upload
Exp		Session label generated by XNAT host is of the form <XNAT Project abbreviation>*<XNAT Subject ID>*<XND Experiment>

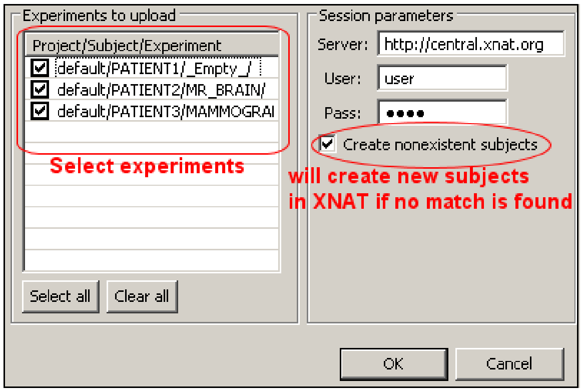
eriment	Uploaded MR session label, see remarks	
Modality	Identifies one of supported session modalities	Currently, only files tagged with "MR" modality will upload successfully.
Scan	Scan label	Displayed in "Scan" column under "Scan Summary" on "Session Summary" page in XNAT. XND DICOM rule tries to extract it from tags (0020,0011) (Series Number) or, if empty, (0020,000e) (Series Instance UID).
Scan_Type	Scan type	Typically, acquisition protocol/sequence description. Displayed in "Type" column under "Scan Summary" on "Session Summary" page in XNAT. XND DICOM rule tries to extract it from tags (0008,0060) (Series Description) or, if empty, (0018,1030) (Protocol Name).

Tags in **red** are required, tags in **green** are optional.

4. Switch to VDS view. You can select entire project, multiple subjects, or individual experiements of selected subjects in the tree view.



5. Select MR sessions for upload (modalities other than MR will be ignored), enter username and password and click OK.



Project/Subject/Experiment
☑ default/PATIENT1/_Empty_/
☑ default/PATIENT2/MR_BRAIN/
☑ default/PATIENT3/MAMMOGRAI

Server: http://central.xnat.org

User: user

Pass: ••••

☑ Create nonexistent subjects

6. Refer to Console view to track the upload progress.

7. On the website, uploaded session files will be listed under Project > Subject > Session, either under "Scan" category (if files were tagged as scans), or "Additional Resources" with custom tags for each file listed.

> NOTE. Currently, uploading files with the same project/subject/experiment combination repeatedly will result each time in re-creating of the session, so previous session data will be deleted. However, you can upload multiple sessions (experiments) for each subject, as long as they have a unique 'Experiment' tag for a given subject.

10. XNAT Desktop Customization

Defining Custom Tag Hierarchies (Ontologies).

XND allows to define custom metadata (which may contain a mixture of category labels, derived measures, data storage hierarchy, experiment attributes, etc.), through attaching the "name-value" pair (XND tag) to each managed file. XND also employs tag hierarchy, or ontology, which reflects child-parent dependencies of concepts used for describing data from a particular domain or subdomain. Originally, XNAT was developed as research neuroimaging data archiving system, and as such it employs the hierarchy of entities (Project->Subject, etc.) to use in neuroimaging research projects.

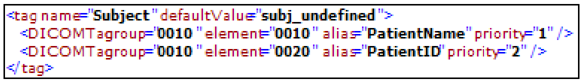
By default, XND is configured with tag hierarchy compatible with XNAT, to allow automatic tagging and batch uploading of arbitrary data to XNAT servers. However, XND can be configured with any other tag hierarchy that meets the needs of your local domain better, and can serve as tag-based database solution for local archiving and network sharing of categorized research data. If you plan to re-distribute XND or use it over a number of different sites, one basic tag system (as defined by local ontology) shared by all users working on the same data domain, can serve as a backbone for such solution.

Tag hierarchy used by XND is defined in View->Preferences->Ontology and rules->Default ontology XML. You can define your own ontology using the same tag definition pattern. Currently, one ontology at a time is supported by XND. For each tag in ontology, only name is required attribute. To show tag in a tree under VDS hierarchical tag view, you can either instruct the program to show it under the VDS tree root ("treeRoot" attribute set to "1"), or list it among descendants of another tree root (use <child> element to define immediate child relationship for a specified tag). The hierarchy you can define may be absolutely arbitrary and contain multiple roots. Use <value> element to add pre-defined values to the tag. You can validate your new ontology xml with the ontology schema xnd_tags.xsd, contained in the same folder, and use 'Test' button at the preference page to see if parcing was successful.

Figure 5 shows VDS view of the same data as on previous screenshots, but using PACS-like ontology, described by this ontology xml document. The tags here were automatically extracted using modified DICOM rule (see next section for creating your rules).

DICOM rule customization.

XND provides a way to create your own rule to generate tags from DICOM files. The default DICOM rule produces a minimal set of tags necessary to display default XNAT ontology. DICOM rule used by XND is defined in View->Preferences->Ontology and rules->DICOM rule XML. For generating each tag described under <tag> element in dicom_rule.xml, XND loops thourgh all <DICOMTag> subelements in order of their priority (defined by "priority" attribute), until non-empty tag is found, or using value specified in "defaultValue" attribute otherwise. For instance, the following inset:

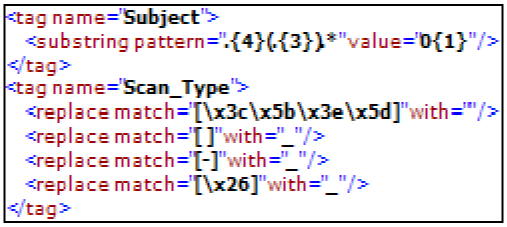


```
<tag name="Subject" defaultValue="subj_undefined">
  <DICOMTag group="0010" element="0010" alias="PatientName" priority="1" />
  <DICOMTag group="0010" element="0020" alias="PatientID" priority="2" />
</tag>
```

describes the creation of "Subject" tag from a DICOM file. First, the program will try to find value of DICOM tag (0x0010,0x0010) in the file, and if empty, try the (0x0010,0x0020) tag, and finally, if both patient name and patient ID tags are undefined (which should never happen as it constitutes severe DICOM violation), will use a "subj_undefined" value for the "Subject" tag.

Tag pattern rule customization.

Tag pattern rule allows two types of operation on tag values: 1) extracting substring and 2) replacing /removing characters from a given character set to a specified character. When constructing your pattern rule, follow the exapmple of the following xml inset:



```
<tag name="Subject">
  <substring pattern=".{4}(.{3}).*" value="0{1}"/>
</tag>
<tag name="Scan_Type">
  <replace match="[\x3c\x5b\x3e\x5d]" with=""/>
  <replace match="[ ]" with="_"/>
  <replace match="[-]" with="_"/>
  <replace match="[\x26]" with="_"/>
</tag>
```

You can combine two types of clauses:

- <substring> instructs the rule to use the first capturing group (i.e. elements caputred in () described in 'pattern' attribute, prefixed and/or postfixed with additional strings according to "value" attribute value. Thus, the rule based on the inset will replace value of tag 'Subject' of type 'nnnnabcd' with strings '0abc'. There can be no more than one <substring> element for each <tag> block.

- <replace> element tells the rule to replace all characters from character set described in 'match' attribute with string contained in 'with' attribute. Replace clauses are executed on the result of substring extraction (if 'substring' element is present in the <tag> block).

> NOTE. As with other rule descriptors, regular expressions used for describing substring pattern and tag match, are based on POSIX regular expression specification implemented in Java 1.5. It might be useful to test your regular expressions with one of java regexp test applets available on the web.

Naming rule customization.

XND provides a way to design a rule to extract meaningful tags based on directory and file names, if the local data directory/file is organized consistently, and file/folder hierarchy and names uniquely identify tags that require extraction. An example of well-defined local structure is OASIS data or Freesurfer output directory hierarchy. To define an XND rule for generating meaningful tags from your files and folders, you'll need to create an XML file where tag generating rules will be described, and select it as an active naming rule in View->Preferences->Ontology and rules->naming rule XML. First, define the folder /file structure: under the <root_dir> element, define all files and folders which the rule may encounter (all others will be ignored). Files/folders are defined by <file> / <folder> elements under <root_dir>, respectively. There can be multiple definitions matching one actual file/folder on the file system.

*Folder element attributes and child elements* include:

- **ID** – unique ID of the folder among all files/folders specified in the XML file;

- **pattern** – specifies the regular expression that identifies this file/folder. If pattern is not specified, ID is used for an exact match. The syntax of the matching pattern corresponds to Java flavor of POSIX regular expressions, as specified here.

- **treeRoot** – whether the folder is in the root from where this rule will be applied on the file system;

- **child** – this element can occur multiple times under <folder> element, each occurrence specifying an ID of child file or folder.
  Any tag value inferred from either file or folder name has to be defined by <tag> element, under <file> or <folder> element, respectively. If the tag is defined under folder, it will be attached to all files in this folder, regardless of an individual file name.

*Tag attributes*:

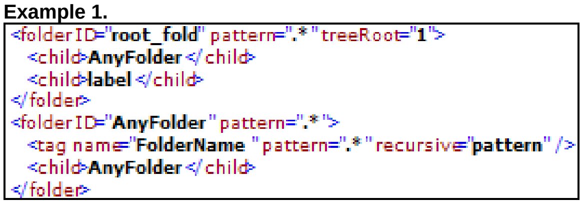
- **name** – the name under which the tag will be stored in XND;

- **value** – if this attribute is present, the tag will have a fixed value specified by it;

- **pattern** – specifies a regular expression with one *capturing group* delimited by () closed brackets. The tag value will be inferred from file/folder name, based on the value matched by the capturing group. See java regular expression specification for more detail on how to construct capturing groups. To verify whether regular expressions and capturing groups do what you intend, you can use this Java applet. Note that 'pattern' attribute is ignored if 'value' attribute is present.

- **recursive** – for tag defined under <folder>, specifies the way of how this rule will be applied recursively to files in subfolders, if any. 'none' prevents from propagating the tag to subfolders; 'fixed' instructs to calculate the tag value based on the current folder's name, and assign the tag with this value to files in all matching subfolders; 'pattern' instructs to re-calculate the value of the tag for each subfolder anew.

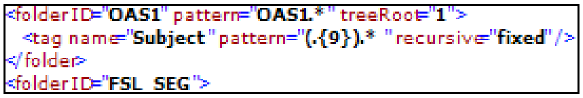
Example 1.



```
<folder ID="root_fold" pattern=".*" treeRoot="1">
  <child>AnyFolder</child>
  <child>label</child>
</folder>
<folder ID="AnyFolder" pattern=".*">
  <tag name="FolderName " pattern=".*" recursive="pattern" />
  <child>AnyFolder</child>
</folder>
```

This inset will tell the program, when applied on an arbitrary folder, to look for all subfolders and apply the tag "FolderName" to all its subfolders recursively.

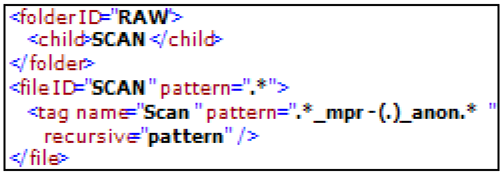
Example 2.



```
<folderID="OAS1" pattern="OAS1.*" treeRoot="1">
  <tag name="Subject" pattern="(.{9}).* " recursive="fixed"/>
</folder>
<folderID="FSL SEG">
```

This folder template describes folders with name starting with “OAS1”, and each file in such folders will be assigned the tag with name “Subject” and value constituting the first 9 characters of the folder name. This tag will be assigned to all subfolders as well. So, for instance, if you have you directory structure with root named after subject, this rule will assign the tag “subject” to all files under this folder, recursively.

Example 3.



```
<folderID="RAW">
  <child>SCAN </child>
</folder>
<fileID="SCAN" pattern=".*">
  <tag name="Scan " pattern=".*_mpr-(.)_anon.* "
    recursive="pattern" />
</file>
```

This inset will result in looking at all files in folder ‘RAW’ and assign tag named “Scan” to all files with name matching regular expression “.**_mpr-._anon.**”, with tag value equal to a literal between ‘.**_mpr-’ and ‘_anon.**’, e.g. file ‘OAS1_0061_MR1_mpr-1_anon.img’ will be assigned tag ‘Scan’ with value ‘1’.