Populate Data

As described in Getting Started / Configuration, the -d data flag is a required parameter to specify a newline-separated text file indicating what projects to import. For convenience, a file called allData.txt is included which will pull all previously prepared sample data. For any project, the script expects two things: a YAML configuration file to define the structure of the project and a folder containing the data for the project. The YAML file must be titled "$projectId_metadata.yaml". The data folder should contain individually zipped sessions at the top level. When the script starts, it will read the text file and begin checking for these files. If the ./data folder does not already contain the YAML file, the script will search for it on the NRG FTP site. Likewise, if the ./data folder does not contain the folder containing the data for the project (or its zip), the script will attempt to locate a zip by the project ID on the NRG FTP site. Below is an example project YAML file which contains most of the currently supported operations (note that the file would necessarily be called study_001_metadata.yaml):

```
id : study_001
runningTitle : First XNAT Study
title : My First XNAT Study
keywords : 1st XNAT
description : This collection contains data for my new XNAT
aliases : [alias1, alias2]
anon :
    file : sample_anon.das
subjects :
    Subj_004 :
        gender : Male
        yob : 1950
        education : 16
        share :
            DEST_PROJ : DEST_SUBJ_LABEL
        complexSessions :
            0000004 :
                share :
                    DEST_PROJ : DEST_LABEL
            empty_session :
                src : empty
                xsiType : "xnat:mrSessionData"
                scans :
                    1 :
                        xsiType : "xnat:mrScanData"
            other_session :
                src : /data/xnat/home/other/session.zip
                assessors : [other_session_QC1]
         assessors : [Subj_004_FORM1]
    GSU002 :
        gender : Male
        yob : 1921
        handedness : Right
        sessions : [GSU002_v00_mr, GSU002_v00_pet, GSU002_v01_mr]
    OUA001 :
        gender : Female
        yob : 1937
        handedness : Ambidextrous
        complexSessions :
            OUA001_v00_mr :
                pipelines :
                    DicomToNifti :
                        create_nii : "N"
            OUA001_v01_mr :
                pipelines :
                    DicomToNifti :
                        create_nii : "Y"
            OUA001_v02_mr :
                scans :
                    3 :
                        xsiType : "xnat:mrScanData"
                        containers :
                            dcm2niix :
                                scan : "/archive/experiments/$sessionId/scans/$scanLabel"
users :
    owners : [owner]
    members : [member]
    collaborators : [collaborator]
pi :
    title : Dr
    firstname : Atticus
    lastname : Cromwell
investigators : [{firstname : Jens, lastname : Aasgaard}, {firstname : Arthur, lastname : Compton}]
accessibility : public
src : /data/xnat/my_project.zip
pluginDependencies : [nrg_plugin_freesurfercommon, nrg_plugin_wmh]
enableWrappers : [dcm2niix]
lastUpdated : 2017-08-28
```

Tips on the YAML above:

anon

The anonymization section mirrors the syntax exactly of how it is used in Site Configuration, except now it is added to the project.

sessions versus complexSessions

The **sessions** element allows you to easily define an array of sessions which will all be uploaded to the XNAT server as is. So, while it is exceedingly easy to specify the sessions as an array like this, you lose power to fine-tune the sessions, as needed. So, if you want to do this, you specify the sessions under **complexSessions**, instead. The sessions here will also be uploaded through the session importer, but additional modifications will be added. So, the study_001 folder for this project should contain the following files:

- `0000004.zip`
- `GSU002_v00_mr.zip`
- `OUA001_v02_mr`
- `GSU002_v00_pet.zip`
- `GSU002_v01_mr.zip`
- `OUA001_v00_mr.zip`
- `OUA001_v01_mr.zip`
- `OUA001_v02_mr.zip`

Note that there are two sessions left out of the above list: "empty_session" and "other_session". If a complexSession has the *src* property set to empty, the session will be created as an empty experiment directly through the experiment API. Then, scans can be manually created and modified under the session. For any scan, such as the above scan 1, you can set *seriesDescription, type, note* in addition to the required *xsiType*. This is the path "empty_session" will take. If a complexSession has the *src* property set to something else, the zip for the session to be uploaded will be searched for at this value (interpreted as an absolute path). So, the "other_session" would have its session zip pulled from /data/xnat/home/other/session.zip.

users

If the users already exist, and the XNAT user running the script has access to the site user list, the users will be added to the project in their corresponding roles. If the users do not already exist, insecure users will be created (the password is just the username), which should not be used in production (this requires admin access).

resources

For any project, subject, experiment, or scan in the YAML, you can add a **resources** block. For project resources, this would be included at the root level, for subject resources, it would be a property of a subject, and so on for complexSessions and scans.

This block should be formatted as:

```
resources :
    resource_folder_1 :
        files : [my_file.txt, my_paper.pdf]
    resource_folder_2 :
        complexFiles :
            my_zip.zip :
                unzip : true
```

The above would create 2 resource folders: one called "resource_folder_1" with the files *my_file.txt, my_paper.pdf*, and one called "resource_folder_2" with the zip *my_zip.zip* extracted. Note that the resources are expected to be in the project's data folder to be picked up for upload. The complexFiles option is similar to the 'complexSessions' option from the experiments section. Files under complexFiles may also specify a 'content' to be added to the REST call to POST them.

src

By default, XNAT Populate will look for the data:

1. First, it checks in a folder under the "data" folder with a name matching the project ID.
2. If that doesn't work, it checks for a zip file under the "data" folder named $project_id.zip.
3. If that doesn't work, it checks on the NRG FTP site for the data.

However, there is another option: by specifying the 'src' value at the top level of the project YAML, the script will attempt to load the project's data zip from the file path given here. Technically, this is checked before the FTP site check, so the script will not fail as long as it finds your data via 'src'. This is useful if the data zip is large and mounted in a reliable spot (it will still be extracted into "data" to post the session zips).

pluginDependencies

Some data requires that your XNAT have certain plugins installed (for example, attempting to import FreeSurfer assessors requires that your XNAT have the FreeSurfer schemas from the FreeSurfer common plugin). As such, this list can be used to specify plugin IDs required by the data in this project. Then, if those plugins are not detected in initialization, populating this project will be skipped.

enableWrappers

This refers to the "uniqueAlias" field in the images section of Site Configuration. When setting up this project, the wrappers identified by the names in this array will be enabled on this project.

lastUpdated

XNAT Populate caches its data in the 'data' folder by default. However, when the configuration for one of the sample projects is updated, XNAT Populate may not be able to find new data that is now required since its project data is cached. Therefore, if 'lastUpdated' is specified with a date, the cached data folder and/or zip for this project will be deleted from the 'data' folder if it's last modified date is **older** than the date specified in this field. This will then allow XNAT Populate to download a new copy of the data from the FTP site.

containers

Note that scan 3 on session OUA001_v02_mr specifies a 'containers' section. The first key, 'dcm2niix' is the uniqueAlias specified in the 'images' section of Site Configuration. This key is mapped to another map (here, it's a simple key-value pair of scan : "/archive/experiments/$sessionId/scans/$scanLabel"). This map will be passed as JSON body to the POST that launches the container on this scan. Currently, XNAT Populate will parse the strings in the map and replace $sessionId with the accession number of the parent session, and $scanLabel with the ID/label for the scan. Note that specifying a container on any object in a project will add a pluginDependency (read the pluginDependencies section for more info) on the Container Services plugin.

What is available already?

See this cheat sheet: xnat_populate_cheat_sheet.md.