Import - Export Test Plan

Purpose

Ensure that the import-export utility meets stakeholder requirements for phase 1:

- Support transacting in RDF
- Support allowing the option to include Binaries
- Support references from exported resources to other exported resources
- Support import into a non-existing Fedora container
- Support export of resource and its "members" based on the ldp:contains predicate
- The URIs of the round-tripped resources must be the same as the original URI

Target Audience

- Fedora committers responsible for developing and maintaining the utility
- Fedora implementers who want to export their resources from a Fedora repository and import them into another repository, including but not limited to another Fedora repository
- Fedora developers who want to customize and extend the utility to meet their organization's particular needs

Components

- Test RDF data sets that express inter-resource relationships and contain Binaries or stable external references
- A small set of fabricated, repeatable Binaries
- Scripts to generate additional RDF data sets
- Checksum comparison tool
- Test HTTP GET requests and SPARQL queries in vagrant environment
- Performance tests and analysis tools

Testing

The ultimate goal of this document is to facilitate the evaluation of the Import and Export Tools utility with respect to:

- Fixity of Binaries
- Graph isomorphism of RDF resources
- Performance

Resources for tests can be found in the following places:

- datasets:
  - Plant Patents
  - LUBM

Concrete Tests to Perform at the Conclusion of Phase 1

1. Using a baseline pre-import/export list of URIs and binary checksums for the plant patents test data, automate the verification of URIs and checksums for all resources post-export/import.
2. Spot-check the RDF graphs of several items, comparing the individual triples for discrepancies (facilitated by conversion of the graph to N-Triples and sorting of the triples). Document the findings.
3. Spot-check the results of a small set of Solr queries performed pre-export and post-reimport, to verify the results. Document the findings.

Long-Term Test Strategy and Goals
Test Cases (which will hopefully get automated in a future phase)

1. **Simple Export**
   a. **Goal**: verify that data is exported to disk
   b. **Setup**: A new, clean fedora4 repository has been created, either manually on a system or via the use of vagrant. Test data has been loaded into that repository. The set should contain both RDF Resources and Non-RDF Resources.
   Steps:
   i. Run the import-export tool on this repository specifying the location in the file system in which the data should be stored.
   d. **Verification**: Spot check the data to verify that binaries and descriptions are represented in the exported data.

2. **Simple Import**
   a. **Goal**: verify that the import-export tool will successfully import data that was previously exported via the tool.
   b. **Setup**: A fedora4 system exists and contains some test data, which includes both RDF Resources and Non-RDF Resources.
   Steps:
   i. Import data from a previously exported dataset using the import-export tool.
   ii. Ensure that the dataset is loaded correctly and that both binaries and descriptions were imported correctly.

3. **Complex Import**
   a. **Goal**: verify that a fedora4 repository that contains complex relationships will produce the equivalent graph when exported and imported.
   Setup: A fedora4 system that contains test data which has external references, cyclic references, and complex ldp:container relationships.
   i. external reference: the dataset contains Resources that have references to objects outside of the repository
   ii. cyclic references: \(<a> ldp:contains <b> or <a> bf:relatedItem <b> and <b> bf:isReferencedBy <a> 
   iii. LDP containers: direct (and indirect for phase 2?)
   Steps:
   i. Export data from a fedora4 system that contains the above described complex relationships
   ii. Import data into a fresh fedora4 system.
   iii. Ensure, via validation tool, that the new system's RDF graph is the same as the old system's.

4. **Import / Export**
   a. Export newly imported data and compare the filesystem trees to ensure that everything looks the same.

5. **Binary Fixity test**