Design for Supporting Test Object Creation in Production

Highlighting indicates what I think are the minimal options for 2.0. It may be that we also want to make some of the other changes at the same time, but if we are concerned about getting them done, they should be delayable.

Highlighting indicates items that have at least an initial implementation (not fully tested, but compiles).

Proposal:

- The Preferences "Purpose" flag currently in use will be extended to support "Testing-Only" (the current value), and/or "Production". Repositories wishing to do both through one profile would add both values (i.e. as a JSON Array)
    - http://seadva-test.d2i.indiana.edu/sead-c3pr/api/repositories/um-arc-ts shows an array with both values
    - http://seadva-test.d2i.indiana.edu/sead-c3pr/api/repositories/bob shows a "Testing-Only" value
- The Staging Area allows the Purpose flag to be set when matchmaking, so for 2.0 there is a mechanism for the user to see the correct list of repos given the purpose. However, since repos may not support the other options, it will now be a potential problem if users change the flag after making a selection, so, post 2.0, we could/should explore not allowing change to Purpose after selecting a repository and/or auto-updating the matchmaking results whenever the flag is changed. (The service layer can check to make sure the purpose is OK w.r.t. the repo selected, but this should redundant if the staging area is making them match to start and the repositories check as they process too.). (Since we are treating purpose as a go/no-go decision unlike other rules, which are only advisory, the proposal is to not list the repositories that can't support the request. Alternately we could cause them to be grayed-out/ use some other visual cue to indicate that the request will definitely fail.)
- Matchmaker will look at the Purpose on the request and in the profiles and determine the match. This will be implemented as a rule. Initially, the matchmaker engine will remove entries that don't meet this requirement. Going forward, we could extend the rule definition to allow rules to be mandatory, so that the Staging Area could show repositories that cannot process the request while still stopping people from sending such requests.
- When a user submits a pub request, it is processed as is currently done by services and repositories with the following changes that depend on the Purpose flag setting:

    Services: The Purpose setting should be added to the summaries provided by api/researchobjects, api/researchobjects/new, api/repositories/<repo_id>/researchobjects and /api/repositories/<repo_id>/researchobjects/new endpoints, and/or add a ?purpose=<x> flag to these endpoints so that repositories can filter the list - doing this filter would probably make it easier for repositories and could reduce the work to get to 2.0.

    Repositories: Repositories should retrieve and process the requests that match their profile "Purpose" setting. (Nominally all requests will have the right setting because Clowder won't send ones that don't match, but switching the profile setting could result in some that don't match, etc. so a quick test is advisable. Repositories should track the purpose of publications and should make the distinction clear to users. *The expectation is, as it is now, that test objects will be processed in a way that will not result in a persistent public identifier and ongoing storage of the data by the repository (will not create something that can be mistaken for a production publication) and in a way that mimics the processing of real objects to some useful degree. How these are defined is up to the repository. (Our reference repository and IU Cloud will end up minting test DOIs that last for two weeks, running some form of cleanup code that removes data from the repository, and will mimic production objects by presenting a temporary landingpage that is similar to or exactly the same as it would be for a production publication. Other repositories could not mint a PID for test objects, create a real ID and mark it as invalid at some later date, perform internal processing but not display a landing page, etc. It is advisable that repositories send status messages back that indicate how processing of the test object did/did not mimic a production request.)*

    Services: When a success message is returned, the service layer will decide, based on the Purpose flag, whether to send FGDC metadata to the production or test DataOne instance.

    Services: As a default, the service layer will remove test objects after 2 weeks. We could consider making this period repository and/or identifier specific, or requiring repositories to send an 'invalid/deleted' status message, and/or have the service layer also mark the request as obsolete (rather than deleting it as is currently done for test objects after 2 weeks). The current implementation, which (I think) triggers off the use of a temporary DOI (recognized by its shoulder) would instead trigger generically off the presence of the Purpose flag having a test value.

    Spaces: Spaces should visually indicate test instances, using the value of the Purpose Flag to do so.It should be clear wherever the returned PID is shown that it should not be used as a long-termID/ that it is temporary. (E.g. SEAD 1.5 marks test requests with a different color and shows a "Temporary DOI" flag next to the ID on test objects in the space's Published Data page.)

    Services: Currently, if the Purpose is test, the API will allow a DELETE on a pub request to succeed even after a repository has started processing the request (non-test requests cannot be deleted once the repository has responded with an initial status message). As a start, we can leave this as is. If, in the future, we want test object requests that cannot be deleted (e.g. by a user in the staging area) once a repo has started working on it, we could adjust this (add another flag). Until then, repositories should be robust against errors (e.g. 404) when they try to submit status for test objects that may have been deleted.

Repository Specific Changes:

Reference Repository:

Repositories using this codebase can work in this mode already (e.g. SEAD Internal, UM-ARC, NDS test) and we will just need to update their profiles to include the appropriate Purpose values() to make them test only, production, or both. One useful update would be to automatically sync the configuration with the profile (right now, whether to generate test or production DOIs is a config option and, if the test flag differs, the code generates a warning log entry and stops/fails and the curator processing the request has to change the config to match to go forward (basically a safety check to avoid creating the wrong type of ID since we currently do not filter requests coming from the space with an incorrect Purpose flag.). The reference repository landing page could also adopt the color/warning notice that the DOI is temporary that exists in 1.5 based on the flag value. To support this without having to store/track the original request, the ref-repo library will update the oremap to indicate that the map and aggregation are for the stated Purpose. It would also make sense to update the ref repo library to positively test for the production flag as well (versus just checking the presence/absence of the test value).

IU Cloud:

Since the IU Cloud repository uses the ref-repo library, it has the basic capability to switch between test and production DOI generation, but, if I understand correctly, it is currently configured as two separate web apps (at http://seadva.d2i.indiana.edu:8081/landing-page/home.html and http://seadva-test.d2i.indiana.edu/landing-page/home.html) that in turn rely on profiles on the production and test instances of the services. (Note the seadva-test profile incorrectly identifies the production IU Cloud as the repository URL in its profile rather than it's own URL). The IU Cloud Search /listing capabilities depend on live api requests to the particular service instance, and the automated publishing and clean-up scripts that are running against the test services do not yet look at the Purpose flag. A short-term option would be for IU Cloud, with minor changes, to continue to run two instances for test and production. Each would have a separate profile, which would need different IDs. Because the requests to each profile would be separate, the search, landing page, test cleanup, automated publishing and other scripts could continue to be configured separately for test and production. A more integrated solution would be one that updates the non-reference parts of IU Cloud to be aware of the Purpose flag. That would allow IU Cloud to run one repository web app for both purposes, have one profile, etc. This should still be relatively lightweight given that the additions would basically need to filter test versus production or highlight them in display(e.g. the search listing of all publications would work the same way except for the need to filter test versus production first, and/or mark test versus production pubs visually.)

Other Repositories:

Other repositories would have the same choice as for IU Cloud - create separate profiles for test and/or production and run different instances of the repo retrieval agent software to manage the requests to those profiles, or create one profile and update the request processing agent to be aware of the Purpose flag. the first option should be minimal work and could be a reasonable 2.0 option. (Specifically, an test repo that wants to be visible to user of 2.0 would need to add the Purpose /testing flag to their profile and submit that profile to the production services instance (changing the ID if they would also have a production profile on the production services.

Interaction with republishing/revision capabilities:

The purpose flag is mostly orthogonal to the republishing mechanism that has been developed. For republishing we have a Preference flag to indicate when to reuse a PID and an 'alternateOf; flag indicating when an earlier publication may be a source of valid metadata/data. The primary interaction with the Purpose flag comes from the fact that both test and production requests will be together in one service instance, and possibly in the same repository which would make it possible to use an alternateOf reference to point to a test pub which could provide a local source of data for a real publication. (e.g. for a large test pub which had to pull files from a space at NCSA, the production pub could retrieve the same files from local test copy (after checking the hash value) rather than having to do another remote http transfer). The logic in the service layer that decides whether a new FGDC profile is needed should continue to work OK after we add testing support (since it relies on the External Identifier preference flag and we won't have test and real pubs with the same PID (by policy in gneral and by design with test DOIs). However, we may want to decide whether alternateOf links between test and production instances should be retained and handled the same way as when they alternateOf link is between production objects (i.e. right now, when one production object replaces another the alternateOf info allows a request for the obsoleted RO to be HTTP redirected to the new RO. This could allow a request for a test object to redirect to a production object that replaces it. Right now this would only be possible for 1.5 publications because Clowder does not support the repub flags yet (since we had no data to republish there, it was not a 2.0 priority), but, if we want this type of provenance between test and production objects going forward we should do a quick test to make sure it works, and, if we want it turned off, we should address it in the services layer. perhaps as the other required updates are being made.