Object Repository Protocol

Introduction

Clients can checkout and commit data to the repository. This design describes how this is done.
It does not (yet) describe security, so for now everybody can do anything.

Protocol

The repository protocol has three commands. The first one is to query the object repositories to check which ones there are and what data they contain. The second and third commands are checkout and commit.

Addressing a specific repository is done by providing two parameters that match with the two parameters that are actually used to register the underlying services. These parameters are:

- customer, which provides the customer ID;
- name, the name of the repository.

So for example, you could address the shop repository for the customer called luminis by specifying those two parameters.

Querying object repositories

This command can be used to exchange information about available object repositories and the versions they contain. Information is exchanged between the central server and client. You ask the server what versions it has, either for a specific repository or for all repositories. The result is a collection of services and range of versions. You can then act on that and ask for versions you don't have.

GET repository/query - returns a full list of repositories and their version ranges;
GET repository/query?customer=luminis&name=shop - returns a version range numbers for a specific repository;
GET repository/query?filter=(customer='luminis') - returns versions for any repository that matches the filter.

About queries

Queries (for log information or entries) come in three forms:

1. Without any filter, you simply get everything.
2. With a filter on customer and name you only get the relevant versions for that repository.
3. With an LDAP filter, where you can filter on arbitrary keys and use compound expressions and pattern matching.

Checking out a version

Here you're asking the server for data. As part of the request, you ask for a specific version for one repository. You must always specify a version, since you must know what version you're checking out to be able to commit it back anyway. This means the usual interaction will be to query first and checkout next.

GET repository/checkout?customer=luminis&name=shop&version=x - returns a stream of data for version x from the repository.

Committing a new version

Committing a new version is done from the client to the server. You can only commit new versions to the "master" repository, so this request will fail when talking to non-master repositories (like a relay server).

POST repository/commit?customer=luminis&name=shop&version=x - Commit a stream of data into the repository. The from version is specified to and given to the repository to check for possible conflicts.