Database for Automation Use Cases

Version 0.1
3rd February 2005

Introduction

These are a couple of "use cases" I have for the CCP4 Database handler, from the point of view of using this for both persistent and volatile storage as part of an automation pipeline. To give some context, the terminology I will use is:

fact – something we know at a given point in time, which may need to be recalled
job – the process of performing some data processing or manipulation
project – a set of related jobs which are probably sequential or related

Use Case 1: Start a Project

To: Begin a project, and record the date at which is was begun.

This should take the "name" of a project and create a new record. Perhaps it should also be possible to pick up an existing project by giving the name and the date. Optionally perhaps just picking up by name should return the latest project of that name.

This should return a handle to the project, so that it can be used to create jobs.

Use Case 2: Begin a Job

To: Create a new job within a project, so that we can store facts pertaining to this job.

The job will have a name, and the time at which this job was started should also be recorded. It is unlikely that a job will need to be recovered to have additional facts attached to it. However, it is likely that fetching facts from an existing job will be necessary.

This should return a handle to the job, so that it can be used to store and recall facts.

Use Case 3: Record and Recover a Facts

To: Record a fact pertaining to a job, and recover it a short or long while later.

To record a fact, which may be a simple string, or a string serialization of a python object via "pickle", associated with a name and a job. If a fact of this name already existed in the database, this should be retained, but the new fact also recorded, along with a time when this was recorded. To recover a fact, the job and name should be given, and the latest example of that fact returned.

There is no need to return a handle to the fact, since there should be a mechanism in place to recover the value of the stored fact anyway.

Here is an example of a “pickled” python object (very small)

```
(i__main__
foo
p0
(dp1
S'a'
p2
I1
sS'b'
p3
I2
sb.
```

I would recommend that the characters used in pickling a python object are not used as special control characters in the database, or a mechanism is provided which can encode this to prevent it being a problem (perhaps base64 encoding would be appropriate, since this is a standard mechanism for encoding binary data in a text file, using only “standard” characters). This will make it look like this:

```
KglfX21haW5fXwpmb28KcDAKKGRwMQpTJ2EnCnAyCkkxCnNTJ2InCnAzCkkyCnNiLg
==
```

I'm not sure if this kind of encoding and so on should be done at the application layer or in the API, but at the moment I think we should assume that it will be done by the application to simplify matters. I don't think that the encoding is computationally expensive, but it will increase the amount of storage required by a small amount (probably less than 50% though).