Experience with Filling a DBI Table

Brett Viren

Physics Department

BROOKHAVEN NATIONAL LABORATORY

Offline Workshop, IHEP, 2011-07-11

Outline

Overview

Standard Operating Procedure

Automate the Filling Process

Service Mapping for Static $\longleftrightarrow$ Dbi

Current Status and Still To Do

Context

Starting point:

- `StaticCableSvc` uses text files, eg `feeCableMap.txt`.
- There are 3 schema for these text files (2 for FEE, 1 for FEC).
- These files have to unrelated maps: sensor-channel and sensor/channel-hardwareID.
- A "Master" text file pretends some features like DBI
- There is also a `DbiCableSvc` that works off the combined schema
- Many hard coding of "`StaticCableSvc`" name in user config code.

Overview of the Steps Taken

Broad steps

- Refactor schema to make two DBI table pairs (`CableMap`, `HardwareID`)
- New ICableSvc implementation (`DybCableSvc`) to front these tables
- Convert and load existing text files into the DBI tables following SOP
- Make a global switch between "static" and "dbi" services in `nuwa.py`
- Change existing code to use generic service name

Details at: `http://dayabay.ihep.ac.cn/tracs/dybsvn/ticket/880`

Which Database

In NuWa, the main database named `offline_db`.
In DBI SOP there is also one named `tmp_offline_db`.
They connect to a real database through `~/.my.cnf`

```
[offline_db]
host            = home2.phy.bnl.gov
user            = reader
#host           = dayabaydb.lbl.gov
#user           = dayabay
password        = XXXXXX
database        = offline_db

[tmp_offline_db]
host            = home2.phy.bnl.gov
user            = bviren
password        = XXXXXX
database        = tmp_bviren_offline_db
```

Standard Operating Procedure for DBI Filling

- Rigorous procedures for updating the master DB.
- Highly controlled, carefully recorded, easily reproducible.
- Well documented in the Offline User Manual[1]
  - Definitions, DBI explanation, software rules, workflow, etc

[1] At `http://dayabay.bnl.gov/oum/sop/` and other locations.

SOP Workflow

Expanded from the SOP workflow[2]

1. Obtain quantities to be inserted into `offline_db` tables.
2. Validate quantities using expert understanding and test codes.
3. Make a temporary (partial) copy of `offline_db` to `tmp_offline_db`
4. Checkout current `tmp_offline_db` CSV catalog from `dybaux` SVN
5. Dump `tmp_offline_db` on top of CSV catalog.
   - Assure no changes!
6. Insert new quantities into `tmp_offline_db`
7. Dump modified `tmp_offline_db` on top of CSV catalog.
   - Assure only expected changes!
8. Commit to SVN and notify DB masters of the pending update.

[2] `http://dayabay.bnl.gov/oum/sop/dbops/#workflow-outline`

SOP Details: Step 3

3 Make a temporary (partial) copy of `offline_db` to `tmp_offline_db`

```
# Dump offline_db to SQL file
db.py offline_db dump offline_db.sql

# Load into tmp_offline_db - drops any preexisting
db.py tmp_offline_db load offline_db.sql
```

Depending on network and size of tables, both steps should take only a few seconds. Future work (S.Blyth) may allow targeting specific table pairs.

SOP Details: Steps 4, 5

4. Checkout current `tmp_offline_db` CSV catalog from `dybaux` SVN
5. Dump `tmp_offline_db` on top of CSV catalog.
   - Assure no changes!

```
# checkout CVS catalog
CSVCAT=http://dayabay.ihep.ac.cn/svn/dybaux/catalog
svn co --username=bv $CSVCAT/tmp_offline_db tmp_offline_db

# Dump tmp DB on top of CSV catalog
db.py tmp_offline_db rdumpcat tmp_offline_db/

# Check for no change
svn diff tmp_offline_db/
```

SOP Details: Step 6

6 Insert new quantities into `tmp_offline_db`

For `CableMap`/`HardwareID` the commit dates of the `fe[ec]CableMap.txt` files and their contents are used as input and a control "`commit.txt`" file drives things.

```
pushd $TESTCABLEMAPROOT/run/
../share/load_static.py commits-r12820.txt [1|2]
```

The `[1|2]` selects Data vs. Simulation for the context. The `commits-r12820.txt` file looks like:

```
r12820 2011-06-30 11:11:45 -0400 feeCableMap.txt | bv | (Thu, 30 Jun 2011)
```

and is constructed via the output of `svn log` for that commit.

SOP Details: Step 7

7 Dump modified tmp_offline_db on top of CSV catalog.

- Assure only expected changes!

```
# Dump on top of CSV catalog
db.py tmp_offline_db rdumpcat /data0/bv/offline/database/tmp_off

# check that correct files were touched
cd tmp_offline_db/
svn status
svn diff LOCALSEQNO/LOCALSEQNO.csv
```

Automation for CableMap/HardwareID

- Package: Database/TableTests/TestCableMap/
  - others should consider adding their own packages under TableTests
- Driving script share/dlfcrs.sh
  - "dump, load, fill, checkout, rdumpcat, sync"
  - ugly, developed while working out the SOP kinks
  - requires re-editing to get different behavior

```
prime_db          # Step 3
prime_svn         # Steps 4 and 5
# load revision of given file for data(1) or sim(2) context
do_one [revision] [file] [1|2]
```

- If more than one revision is loaded in a single filling session the first two priming functions should be commented out.
- Each do_one is run and then commented out and the SVN commit revision is noted to provide a record of what has been loaded.

Service Mapping

In DybPython.Control.NuWa (ie, nuwa.py) new utility method:

```
def add_service_redirect(self,alias,name):
    """
    Make alias an alias for given service.
    Should be called during configuration only
    """
```

If DBCONF is defined or nuwa.py --dbconf= option given switch between static and (some) DBI versions of the data service trio:

- CableSvc (StaticCableSvc vs. DybCableSvc)
- SimDataSvc (StaticSimDataSvc only)
- CalibDataSvc (StaticCalibDataSvc only)

Not all have DBI versions yet, update the switch as they are developed. Will need to be extended for any novel services developed.

Migrating existing code

There exists many places that use explicitly the name "StaticCableMap"

- Default property values
- Job modules
- Hard coded

Most have now been fixed to use the generic "CableMap".
Did not touch code that used something besides the standard `feeCableMap.txt`
Converted some, but not all, of SimDataSvc and CalibDataSvc.

NB: Hard Coded Local Paths

During the migration I found a few cases of hard coding local path! Many were fixed by the migration but one should instead rely on the many environment variables that NuWa gives - at least one for each package!

```
# bad
CableMapFile = "/afs/ihep.ac.cn/soft/dayabay/NuWa/daily"+\
               "/release/NuWa-trunk/dybgaudi/DataModel"+\
               "/DataSvc/share/feeCableMap.txt"
# better
import os
dsroot = os.environ.get("DATASVCROOT")
CableMapFile = dsroot + "/share/feeCableMap.txt"

# best
# ... nothing here, since the nuwa.py now handles this
```

Current Status

- Migration to a SOP-compliant DBI tables for CableMap/HardwareID done.
  - Future updates probably best done by committing static table and repeating the `dlfcrs.sh` procedures.
- New `DybCableSvc` added to new `DybSvc/DybMetaDataSvc/`.
  - Recommend all SOP-compliant DBI services (at least among the trio of data services) be added to this package.
- Single Static/DBI switch added to `nuwa.py`.
- Most C++ and Python code modified to use generic `CableSvc` name.

Still To Do

- SimDataSvc will get the treatment next.
  - I leave CalibDataSvc to the calib experts but am available to help.
- Obsolescence of DbiCableSvc and StaticCableSvc
  - Is there a reason to keep StaticCableSvc in the long run?
- Move/remove DataModel/DataSvc and Database/DbiDataSvc
  - Irrespective of the migration they better belong in DybSvc/
  - I would like all new SOP-compliant services are put in DybSvc/DybMetaDataSvc so we can remove DbiDataSvc
  - Remove DataModel/DataSvc if we no longer want the static counterparts.