
BLAST CAOM Creation Logic

Version 1.0
2008-04-08

**Russell Redman,
National Research Council of Canada
Herzberg Institute of Astrophysics
Canadian Astronomy Data Centre**



Table of Contents

1	Creation versus re-use of CAOM Objects	2
2	Order of Ingestion of BLAST files	3

1 Creation versus re-use of CAOM Objects

Use an existing Observation if
 collectionID matches OBSID
otherwise create a new Observation.

Use an existing Plane and Output (note the 1-1 relationship) if
 Plane->Observation.collectionID matches OBSID
 Plane->Output.name matches PRODUCT
 Plane->Output.version matches VERSION
otherwise create new Plane and Output objects.

Use an existing Process if
 if PRODUCT in (“REDUCED”, “NOISE”, “HITS”)
 foreach Output o in Process.OutputList
 o->Plane->Observation.collectionID matches OBSID
 o.name in (“REDUCED”, “NOISE”, “HITS”)
 o.version matches VERSION
 else if PRODUCT in (“DECONVOLVED”, “DECONVOLVED NOISE”)
 foreach Output o in Process.OutputList
 o->Plane->Observation.collectionID matches OBSID
 o.name in (“DECONVOLVED”, “DECONVOLVED NOISE”)
 o.version matches VERSION
otherwise create a new Process.
Add the current Output to its OutputList.

Matching Process.name to the FITS header RECIPE would be expected as part of the logic for creating a new Process, and possibly matching process.version to ENGVERS, but is not necessary here because the BLAST team have agreed to change VERSION if they use a different process to reduce the data.

Foreach FITS header PRVn, n=1..PRVCNT
 If there exists an Artifact in CAOM such that Artifact.data = AdFile(‘blast’, PRVn)
 if there does not exist Input in Process.InputList with Input.planeID = Artifact.plane.getID()
 create a new Input with Input.planeID = Artifact.plane.getID()

Note that this will fail, causing the ingestion to fail, if the Input Plane does not already exist. See the next section for a solution suitable for BLAST.

Use an existing MetaReadAccess if
 getDate() < RELEASE
 MetaReadAccess.planeID matches Plane.planeID
otherwise create a new MetaReadAccess.
(Not needed for initial release.)

Use an existing DataReadAccess if
 getDate() < RELEASE
 DataReadAccess.planeID matches Plane.planeID

ERROR! REFERENCE SOURCE NOT FOUND.

otherwise create a new DataReadAccess.
(Not needed for initial release.)

Use an existing Artifact if

Artifact->Plane->Observation.collectionID matches OBSID

Artifact->Plane->Output.name matches PRODUCT

Artifact->Plane->Output.version matches VERSION

Artifact.data matches AdFile("JCMT","FILEID")

otherwise create a new Artifact.

2 Order of Ingestion of BLAST files

The creation of Input objects in the Provenance will fail if the Input planes do not already exist. There is a dependency of the Output Planes on the existence of Input Planes.

For BLAST the dependency rules to ensure that all inputs are ingested before all outputs can be quite simple, because we have a strong assurance from the BLAST team that inputs and outputs will be generated and ingested in complete batches. If we ingest all REDUCED and NOISE products before all DECONVOLVED and "DECONVOLVED NOISE" products, then we can be sure that the input dependencies will be satisfied when the provenance objects are created. This can be handled with a very simple make file.

The directory tree in use at the CADK has no place for a random make file, but an easy workaround is to put a perl script in the scripts directory that writes a suitable make file into the current directory. I have created such a script called makeBlastIngest.pl:

```
#!/usr/bin/env perl
if ( -e blastIngest.make ) { unlink "blastIngest.make";}

open MAKEFILE, ">blastIngest.make";
print MAKEFILE "
REDUCED      := \$(patsubst %.fits,%.fits.done,\$(wildcard *reduced*.fits))
NOISE        := \$(patsubst %.fits,%.fits.done,\$(wildcard *noise*.fits))
HITS         := \$(patsubst %.fits,%.fits.done,\$(wildcard *hits*.fits))
DECON        := \$(patsubst %.fits,%.fits.done,\$(wildcard *decon*.fits))
DECONNOISE   := \$(patsubst %.fits,%.fits.done,\$(wildcard *deconnoise*.fits))

.PHONY: all
all:         \$(REDUCED) \$(NOISE) \$(HITS) \$(DECON) \$(DECONNOISE)

.PHONY: clean
clean:
\t/bin/rm *.done

%.fits.done: %.fits
\t\t@echo doing \${<
\t\ttouch \${&@}
";
close MAKEFILE;
exit;
```

This make file records the ingestion of each fits file by touching an empty %.fits.done file. Obviously, this is

ERROR! REFERENCE SOURCE NOT FOUND.

just a test version of the real makefile, which would run blast2caom.py instead of echoing a message to sysout.

It would be slick, but not necessary for BLAST, to have a script that wrote real file-by-file dependencies into the make file by reading the FILEID and PRVn headers form each product.