# BLAST Archive ICD

Version 1.0
2008-03-26

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

## Table of Contents

## 1 INTRODUCTION

BLAST is a hosted archive, meaning that the archived data will arrive in a few batches that can be ingested with a script from the command line.

BLAST had two flights, being launched once from Sweden and once from Antarctica. A number of targets were observed on each flight. The data from each target have been combined into a single set of images at three different wavelengths (250, 350 and 500 microns). Observations at all three wavelengths were made simultaneously. This is similar to SCUBA scan maps, in which the telescope was also scanned over the target region while gathering data simultaneously at multiple wavelengths.

## 2 OBSERVATION-LEVEL OBJECTS

BLAST data is fully processed before it is archived at the CADC. The archive will not contain "raw" data, nor are small observations combined into larger composite observations. All observations can therefore be treated as simple observations. A simple observation for BLAST will comprise the fully reduced data files for a single target region as well as the products derived from that data. In addition to its own fields, a simple observation should have arguments for

Target (optional)
Telescope
Instrument

Each observation has been assigned a unique OBSID value by the BLAST team, so all FITS files with the same OBSID value in their headers should be grouped into the same observation. In principle, observations are defined by target region on the sky and the OBSID can be considered as an internal nickname for the target region.

At the moment, there is only one observation within the BLAST dataset, for the Vulpecula region, but more will be created as the data reductions are finished.

### 2.1 Arguments for Target

- target.name : from FITS header OBJECT
- target.classification : supplied manually as an argument to the jython script?
- target.redshift : always left NULL

The redshift is left NULL because there will be multiple objects in each field at unknown redshifts.

### 2.2 Arguments for Telescope

The telescope was attached to a balloon and moved a considerable distance during the observations. There is no sensible "average position", nor is the positional and energy WCS sufficiently precise to warrant corrections based on the observatory position. The geoLocation fields are therefore all set to 0.0.

- telescope.name : always "BLAST"
- telescope.geoLocationX : 0.0
- telescope.geoLocationY : 0.0
- telescope.geoLocationZ : 0.0

### 2.3 Arguments for Instrument

| | |
|---|---|
| • instrument.name | : always "BLAST" |

### 2.4 Arguments for SimpleObservation

| | |
|---|---|
| • collection | : always "BLAST" |
| • collectionID | : from FITS header OBSID in the "reduced" file |
| • project | : always "BLAST" |
| • data | : (filled from artifact using caomlib) |

(plus the Target, telescope and Instrument objects created above)

## 3 PLANE-LEVEL OBJECTS

Within a simple observation, planes will distinguish different kinds of products generated by the data reduction pipeline. For BLAST the different kinds of products are:

| | |
|---|---|
| • reduced | : the basic, fully reduced product from which other products are derived |
| • noise | : the noise image associated with the reduced product |
| • hits | : the number of measurements contributing to each pixel in the reduced product |
| • decon | : an image derived from the reduced image by deconvolving with the beam profile |
| • deconnoise | : the noise estimated for the decon product |

All of these are FITS files. The plane appropriate for each FITS product can be determined from the RECIPE and PRODUCT headers that will fill the caom_process.name and caom_output.name fields, respectively.

Observations from the Swedish flight will include the decon and deconnoise files. Observations from the Antarctic flight will not.

### 3.1 Arguments for Observable

| | |
|---|---|
| • ctype | : always "phot.flux.density.sb" |
| • cunit | : from FITS header "BUNIT" |

### 3.2 Arguments for Shape

(filled from artifacts using caomlib)

### 3.3 Arguments for EnergyInterval

(filled from artifact using caomlib)

### 3.4 Arguments for TimeInterval

(filled from artifact using caomlib)

### 3.5 Arguments for PolarizationList

(filled from artifact using caomlib)

### 3.6 Arguments for Plane

| | |
|---|---|
| • metaRelease | : now() or fill from FITS header RELEASE |
| • dataRelease | : now() or fill from FITS header RELEASE |
| • project | : always "BLAST" |
| • data | : (filled from artifact using caomlib) |

### 3.7 Authorization

There are two possible approaches to access control. The first is to make all data public immediately. This is the default if no keyword RELEASE is supplied. The second is to restrict access to the BLAST team and CADC staff initially.

#### 3.7.1 Public access

In this option, there would be no need to create MetaReadAccess or DataReadAccess tables, and the release dates in the Plane object could always be set to now().

#### 3.7.2 Proprietary access

In this option, a release date would have to be supplied by the BLAST team using the FITS keyword RELEASE with a value in the format YYYY-MM-DD. This release date will be used to fill the Plane.metaRelease and Plane.dataRelease fields.

##### 3.7.2.1 Arguments for MetaReadAccess

For proprietary access, there will be two rows in MetaReadAccess
groupID : for BLAST team

groupID : for CADC

##### 3.7.2.2 Arguments for DataReadAccess

For proprietary access, there will be two rows in DataReadAccess
groupID : for BLAST team

groupID : for CADC

### 3.8 Provenance

The provenance comprises three linked objects for the process that created the the files in the Plane, the Outputs produced by the process (i.e. the kinds of files), and the Inputs required for the Process. The Input and Output point to planes in the archive, so data that is not being archived (e.g. BLAST raw data) cannot be represented in this model.

There will be a unique instance of Output for each Plane within the Observation. Every BLAST observation will have three Planes corresponding to the Outputs REDUCED, NOISE and HITS. Since the archive does not hold raw BLAST data, there are no Inputs attached to the Process containing the REDUCED, NOISE and HITS Outputs. Observations can optionally include the Outputs DECONVOLVED and “DECONVOLVED NOISE”, both of which will depend upon REDUCED and NOISE as Inputs.

Because the Input and Output objects take planeID values in their constructors, it is not possible to build the provenance objects until after all of the required Planes have been created in CAOM.

### 3.8.1 Arguments for Process

| | |
|---|---|
| • name | : from FITS header RECIPE |
| • version | : from FITS header ENGVERS |
| • reference | : from FITS header REFERENC |
| • producer | : from FITS header PRODUCER |
| • lastExecuted | : from FITS header DATE |

Every BLAST Observation will have one instance of Process for the REDUCED, NOISE and HITS planes. Observations can optionally have an instance of Process for the DECONVOLVED and "DECONVOLVED NOISE" planes.

### 3.8.2 Arguments for Output

| | |
|---|---|
| • name | : from FITS header PRODUCT |
| • planeID | : the planeID for this product |
| • version | : from FITS header VERSION |

The Output list for a Process cannot be constructed until the CAOM Planes have been created, because they use the planeID of the Output Plane.

Every Plane must be associated with a unique instance of Output and vice versa, i.e. there is a one-to-one relationship between Plane and Output objects.

### 3.8.3 Arguments for Input

| | |
|---|---|
| • planeID | : looking up the planeID for FITS header PRVn |

The Input list for an Output plane cannot be constructed immediately upon ingestion because it cannot be known whether the Input planes have been created yet. The ingestion script must take the list of input files given by the FITS headers PRVCNT and PRV1..PRV<PRVCNT>, look up the corresponding set of planeID values, and insert that set into the Input list.

## 4 ARTIFACT-LEVEL OBJECTS

Within a plane in a simple observation, there is a one-to-one relationship between data files and artifacts. Artifacts are nominally distinguished by their position, time, and energy WCS entries.

### 4.1 Arguments for Position

| | |
|---|---|
| • coordsys | : from FITS header RADESYS (or RADECSYS, default = "GAL") |
| • equinox | : from FITS header EQUINOX (default 2000.0) |
| • ctype1 | : from FITS header CTYPE1 |
| • ctype2 | : from FITS header CTYPE2 |
| • cunit1 | : from FITS header CUNIT1 |
| • cunit2 | : from FITS header CUNIT2 |
| • naxis1 | : from FITS header NAXIS1 |

| | |
|---|---|
| • naxis2 | : from FITS header NAXIS2 |
| • crpix1 | : from FITS header CRPIX1 |
| • crpix2 | : from FITS header CRPIX2 |
| • crval1 | : from FITS header CRVAL1 |
| • crval2 | : from FITS header CRVAL2 |
| • cd11 | : from FITS header CD1_1 |
| • cd12 | : from FITS header CD1_2 (default 0.0) |
| • cd21 | : from FITS header CD2_1 (default 0.0) |
| • cd22 | : from FITS header CD2_2 |
| • crder1 | : always left NULL (can we fill in an estimate?) |
| • crder2 | : always left NULL (can we fill in an estimate?) |
| • csyer1 | : always left NULL (can we fill in an estimate?) |
| • csyer2 | : always left NULL (can we fill in an estimate?) |

## 4.2 Energy

The Energy WCS is not given explicitly in the FITS headers, but is identified as one of three bandpass filters using the FILTER keyword (similar to SCUBA/SCUBA2). There will be three corresponding entries in the energy table for BLAST. The following table gives approximate wavelengths in microns (topocentric) at which the transmission drops to 10% of the peak value, derived from the filter profiles measured for the Swedish run for the wavelengths:

| **bandpassName** | **cval1** | **cval2** |
|---|---|---|
| 250 | 210 | 302 |
| 350 | 295 | 425 |
| 500 | 417 | 624 |

The value of the FITS header FILTER can then be used to select the appropriate row by comparison with the bandpassName.

A row in the Energy table is defined by three objects,

SpectralFunction
SpectralWCS
Energy

### 4.2.1 Arguments for SpectralFunction

There are three instances of SpectralFunction:

| | |
|---|---|
| • naxis | : 1 |
| • crpix | : 0.5 |
| • crval | : 210.0 |
| • cdelt | : 92.0 |

| | |
|---|---|
| • naxis | : 1 |
| • crpix | : 0.5 |
| • crval | : 295.0 |
| • cdelt | : 130.0 |

| | |
|---|---|
| • naxis | : 1 |

| | |
|---|---|
| • crpix | : 0.5 |
| • crval | : 417.0 |
| • cdelt | : 207.0 |

### 4.2.2 Arguments for SpectralWCS

All three rows can use the same arguments to initialise the three SpectraWCS object:

| | |
|---|---|
| • specsys | : "TOPOCENT" |
| • ssysobs | : "TOPOCENT" |
| • ctype | : "WAVE" |
| • cunit | : "um" |
| • crder | : NULL |
| • csyer | : NULL |
| • restfreq | : NULL |
| • restwave | : NULL |
| • velosys | : NULL |
| • zsource | : NULL |
| • ssyssrc | : NULL |
| • velang | : NULL |

(plus one of the SpectralFunction objects created above)

### 4.2.3 Arguments for Energy

There will be three Energy objects corresponding to the three SpectralFunction objects created above.

| | |
|---|---|
| • bandpassName | : from FITS header FILTER |

## 4.3 Arguments for Time

The time WCS is not given explicitly, but can be derived from the approximate start and end UT dates recorded in the FITS headers DATE-OBS and DATE-END. These dates implicitly refer to the time 00:00:00.00, and need to be converted to MJDs for use in CAOM. Refering to the MJD values as mjdObs and mjdEnd, respectively, we can fill in a Time object as a TemporalFunction.

| | |
|---|---|
| • ctype | : always "MJD" |
| • cunit | : always "d" |
| • crder | : always left NULL |
| • csyer | : always left NULL |

## 4.4 Arguments for TemporalFunction

| | |
|---|---|
| • naxis | : always 1 |
| • crpix | : always 1.0 |
| • crval | : (mjdEnd + mjdObs)/2 |
| • cdelt | : (mjdEnd - mjdObs) |

## 4.5 Arguments for Polarization

The Polarization table needs only a single entry for BLAST, indicating that the observations were made of Stokes I.

| | |
|---|---|
| • naxis | : always 1 |
| • crpix | : always 1 |
| • crval | : always 1 |
| • cdelt | : always 1 |

This single entry should be preloaded. The script should locate this row and remember its id, which will be used to fill the Artifact polarizationID.

### 4.6 Arguments for Artifact

| | |
|---|---|
| • naxis | : always 2 (but first verify that FITS header NAXIS = 2) |
| • partNumber | : always 0 (only ingest the primary HDU) |
| • partName | : always NULL (nominally a file name within a tar or zip file) |
| • positionAxis1 | : always 1 |
| • positionAxis2 | : always 2 |
| • energyAxis | : always left NULL |
| • timeAxis | : always left NULL |
| • polarizationAxis | : always left NULL |
| • data | : an AdFile object for the FITS file name |

(plus the ID's of the Position, Energy, Time and Polarization objects created above)

## 5 Filter Profiles

The BLAST web page allows the user to download the filter transmission profiles for each wavelength and observing run. There will be six of these files in total, all quite small. The file containing the filter profile for each artifact will be named in the FITS header PROFILE.

The filter profiles are calibration data and in a larger archive would belong in a calibration database that would be part of the archive-specific metadata (ASMD) rather than CAOM. Since there will only be six of these files, it does not seem worth defining a database table linking each artifact to the associated profile. An alternative approach would be to add a single button to the search page that would pass the names of all six files to downloadManager. The user could then select which of the profiles to actually download.

## 6 Versioning and the creation of Observations, Planes and Artifacts

Since there is only one CAOM collection in the BLAST archive, with the value "BLAST", Observations are uniquely identified by the collectionID which is filled from the OBSID header of a BLAST file upon ingestion. If an existing Observation already has that value for the collectionID, it will be used to record the file. A new Observation will be created if and only if no Observation already in the archive has that value for the collectionID. See section 2.4.

Every Plane in the BLAST archive will be associated with a unique Output in the Provenance tables, i.e. there will be a one-to-one relation between rows in the CAOM Plane table and Provenance Output tables. Within an Observation, every Plane will be uniquely identified by the Output.name field (filled from the FITS PRODUCT header) and the Output.version field (filled from the FITS VERSION header). A new Plane and Output pair will be created if and only if an existing Plane and Output pair cannot be found in the Observation with the values for Output.name and Output.version given by the FITS headers PRODUCT and VERSION respectively.

The BLAST team will ensure that all files with the same values for the FITS headers OBSID, PRODUCT and VERSION contain consistent physical metadata in the FITS headers used to fill fields at the Observation and Plane levels (see sections 2 and 3). When a file is ingested into an existing Observation and Plane, these metadata will overwrite the previous values.

The user interface (UI) provided for the BLAST archive by the CADC will allow users to select different versions of each file based on the Output.version field. (The mechanism of version selection in the UI is still TBD, but if it allows a "default" version to be specified, then the most recent version will be the default.)

Each BLAST file will be represented by a single CAOM Artifact and metadata will be ingested only from the primary HDU, even if multiple extensions are present. Each file will be uniquely identified by the FILEID keyword, which will contain the file name minus the ".fits" extension. The value of FILEID will be used to construct the Artifact.data URI and will be used as the primary key in the blast_entry table that can be used to quickly find file-related metadata in the CAOM tables. A new Artifact will be created if and only if an Artifact with the same value of FILEID (equivalently Artfact.data or blast_entry.entry) does not already exist.