

**SCUBA-2 DR Pipeline Project Office**

University of British Columbia  
 6224 Agricultural Road  
 Vancouver, British Columbia  
 CANADA  
 V6T 1Z1

Tel: +1-604-822-2211

Fax: +1-604-822-5324

Email: [jmolnar@ubc.ca](mailto:jmolnar@ubc.ca)


WWW: <http://scuba2.jach.hawaii.edu>

**Document Title: Pipeline Architecture**

**Document Number: SC2/SOF/S210/001**

**Issue: 1.25**

**Date: 2005/05/24**

Document Prepared By:	Tim Jenness	Signature and Date:	2005-05-24
Document Approved By:	Douglas Scott	Signature and Date:	2005-05-25
Document Released By:	Janos Molnar	Signature and Date:	 2005-05-25



# Contents

<b>1 Introduction</b>	<b>3</b>
<b>2 Overview</b>	<b>3</b>
<b>3 Algorithms</b>	<b>3</b>
<b>4 Data Formats</b>	<b>5</b>
<b>5 Modifications required for SCUBA-2</b>	<b>5</b>
5.1 DRAMA support . . . . .	6
5.2 Data monitoring . . . . .	6
5.2.1 DRAMA monitoring . . . . .	6
5.2.2 Flag file . . . . .	7
5.3 Observation and sub-scan numbers . . . . .	7
5.4 Header Translation . . . . .	8
5.5 Recipe switching . . . . .	8
5.6 Recipe parameters . . . . .	8



# 1 Introduction

This document describes the Pipeline infrastructure. This includes details of the Pipeline system and message busses but does not include any discussion of algorithms.

**Req. GR6**  
**Req. OR7**

The SCUBA-2 DR Pipeline will use ORAC-DR<sup>8,19,1</sup> as its core pipeline infrastructure. This is the standard JAC pipeline infrastructure already in use at JCMT for SCUBA<sup>20,13,21</sup> and, shortly, for ACSIS<sup>18,12</sup> and at UKIRT<sup>7</sup> for UFTI,<sup>6,5</sup> CGS4,<sup>11</sup> MICHELLE, UIST<sup>27</sup> and WFCAM.<sup>24,25</sup> It is also in use for IRIS-2 at the AAT<sup>3</sup> and can be used for the reduction of Gemini<sup>3</sup> and ESO data. The Pipeline is inherently a data-driven design. It is also easily extendable by off-line users and has an open source licence. This choice simplifies the integration of the Pipeline into the current JAC infrastructure. ORAC-DR currently runs on Linux, SPARC Solaris, Mac OS X and TRU64 Unix.

**Req. GR2**  
**Req. GR13**  
**Req. OR8**  
**Req. GR10**  
**Req. GR4**  
**Req. GR5**  
**Req. GR1**  
**Req. GR20**

## 2 Overview

As can be seen from the above, ORAC-DR is a proven truly generic pipeline design. This is possible because the mechanics of a pipeline (for example, detection of new data, reading of header information, determination of observation grouping, dynamic recipe construction and provision of messaging interface) are distinct from the actual data processing steps. This approach can be seen in the architecture diagram, Fig. 1.

The Pipeline itself is written in object-oriented Perl, with the data reduction Recipes written in an abstracted high-level meta-language consisting of primitive steps. The primitives themselves are written in Perl and so the Pipeline reads the Recipe (using the Recipe name either specified explicitly in the FITS header or inferred from the FITS header), converts it to Perl and executes it, such that Pipeline state is available to the Recipe and the Recipe can set Pipeline state. This ability for the Pipeline state to be available within the Recipe is a key advantage of this system over creating a shell script that is simply forked from the Pipeline process to reduce each file.

## 3 Algorithms

Whilst the Recipes concentrate on reduction control, the bulk of the algorithmic data processing is performed using separate “algorithm engines”. These tasks are started by the Pipeline on-demand and controlled via the use of a messaging system (this approach was *not* used for the WFCAM pipeline.<sup>24</sup> In that case they had pre-existing code directly interfaced to Perl using the XS<sup>17</sup> mechanism. The algorithms were written in C but no messaging system was used).

This approach allows the algorithms to be implemented independently of the Pipeline itself. Additionally, it is vital that meaningful status is returned by these tasks, either through an exception or via status return, so that the Pipeline can respond to errors in a meaningful manner. For example, image registration may fail for a number of reasons and it is useful to be able to distinguish between a fatal error (e.g. disk filling up) from an error that can be worked around (e.g. unable to determine enough targets from the data).

Since the Pipeline architecture is in Perl, the only constraint on the type of messaging sup-



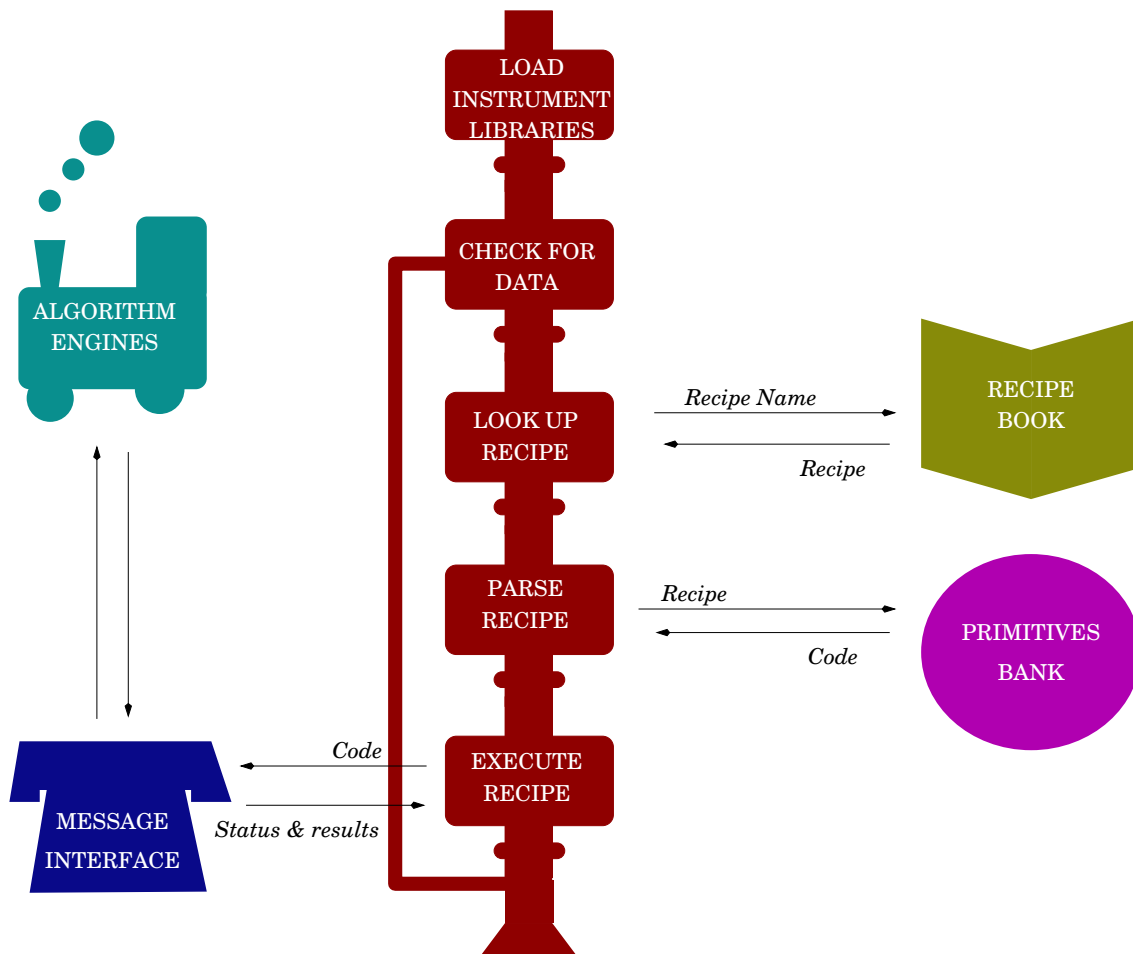


Figure 1: Global Pipeline data flow (diagram from<sup>8</sup>).

ported in order to control the tasks is that this is possible from Perl. The Pipeline itself is flexible enough to support any messaging scheme so long as a single message is sent to the task and a return status made available on completion. The Pipeline currently can use ADAM<sup>23,4,22</sup> and DRAMA<sup>2</sup> message systems.

The Recipe analysis<sup>16</sup> has identified a number of pre-existing algorithm engines that can be used (mainly from Starlink and existing SCUBA ORAC-DR Recipes). Additionally, brand new tasks must be written and they should conform to the standard algorithm engine interface.<sup>15</sup> New code written as part of this project will be written in C and use the DRAMA messaging system as specified in the initial pipeline infrastructure proposal.<sup>26</sup> The FTS software team are providing a DRAMA interface to their Java algorithm engine.

**Req. ER8**

Care will be taken to separate the core numeric processing from both the data access layer and the message processing layer. A generic architecture will be written to achieve this goal.

## 4 Data Formats

The data written by the data acquisition system will use the Starlink N-Dimensional Data Format (NDF)<sup>28,14</sup> and this will be the data format used internally as an intermediate file format wherever possible. File format conversions within primitives are possible if the algorithm engine requires it, but the file must be back in NDF when the particular primitive completes. Repeated file format conversion should be minimized. If it transpires that there are many format conversions within a Recipe it may be necessary to enhance the ORAC-DR object model, such that the current file format is retained and the file can be format converted on request.

All new algorithms engines will be written to use NDF although the file I/O will be separated from algorithmic control, such that the format can be changed at a later date. This does imply that care should be taken to restrict the requirement for hierarchical file structures since not all file formats are hierarchical. The FTS software team is already doing this; their Java algorithm engine uses the Starlink HDX java library<sup>10,9</sup> to abstract file I/O.

## 5 Modifications required for SCUBA-2

The steps involved in adding a new instrument to ORAC-DR are well documented,<sup>19</sup> but can be summarised as:

1. Creating a new `Frame` class containing the knowledge of the file-naming convention, how to read data headers, how to handle sub-images and how to translate headers from the SCUBA-2 values to the standardised internal form.
2. Creating a new `Group` class for operations involving group handling.
3. Creating a new `Calib` class, providing methods for determining optimal calibrations (using the built-in calibration indexing system). Note that filing calibrations is a function of the Recipes.
4. Creation of configuration/start-up scripts.



5. Provision of necessary messaging interface to remote algorithm engines if they do not currently exist (discussion of algorithm engines is deferred to document<sup>16</sup>).
6. Addition of new Recipes and primitives detailing how to actually process SCUBA-2 data. This is the largest part of the Pipeline work (along with the actual algorithm implementation) and is discussed in the Recipes document.<sup>16</sup>

The SCUBA-2 pipeline will actually result in two new instruments from the infrastructure viewpoint. `ORAC_INSTRUMENT` will be set to `SCUBA2_LONG` for 850 microns and `SCUBA2_SHORT` for 450 microns. The only difference between these is in the file name prefixes that they will use (`s8` vs `s4`).

## 5.1 DRAMA support

In order to support the FTS and the new tasks written for the SCUBA-2 project, it has been necessary to extend ORAC-DR to support DRAMA in both the data detection loop (see later) and the primitive layer itself. DRAMA tasks can be started on demand by the pipeline in the same manner as used for ADAM:

```
$Mon{surf2}->obeyw("extinction", %args);
```

Additionally, since DRAMA does not natively treat parameters as a single string of key=value pairs the message handling will be extended to support both "par1=val1 par2=val2" and a hash for arguments to the `obeyw` method. The standard ADAM string form will be converted to a DRAMA `Arg` structure (automatically using `Argument1` and `Argument2` etc if no parameter is explicitly defined. The ADAM interface will be upgraded to support a hash form.

## 5.2 Data monitoring

SCUBA-2 differs from all previous ORAC-DR-supported instruments in that data are written by multiple acquisition computers, with each DA computer providing its own completion notification, but data from multiple computers must be combined (4 sub-arrays at once, one pipeline per wavelength).

**Req. PR15**

As described in the ICD<sup>14</sup> each DA-computer uses two schemes for data notification. The first (for QL mode) is to update a DRAMA parameter with the most recent (processed) data set. The second (for the science pipeline) is to write a flag file to disc containing the names of all the files written for that observation.

The pipeline infrastructure has been extended to support both these changes.

### 5.2.1 DRAMA monitoring

In Quick Look mode the pipeline must look at a DRAMA parameter on each of the 4 data acquisition computers (for the specific wavelength) to determine whether new data are available.



A new looping option (`-loop task`) can be used to monitor data from a remote parameter. In this mode, the pipeline reads the structured parameter value from each of the 4 data acquisition computers (which may be a pointer to a file or an image with a FITS header, see SC2/SOF/IC210/01 for details of the format) and, for the image case, writes the data values and FITS header to disc in a format suitable for reading by the SCUBA-2 Frame classes. The pipeline then processes the data as for any other data file.

This loop option has been implemented in a message system agnostic manner such that any message system can be used that supports hierarchical parameters. The SCUBA-2 Frame class specifies the DRAMA messaging system.

As described in the ICD the parameter includes a sequence number. This sequence number allows the pipeline to indicate to the user when a frame has been missed whilst guaranteeing that the most recently obtained data are always shown as quickly as possible.

**Req. DR8**

### 5.2.2 Flag file

Existing ORAC-DR implementations support a single flag file containing the relevant input files (this feature was added for ACSIS<sup>18</sup>) although most instruments rely on a single zero-length flag file (e.g. CGS4 where the single file contains multiple readouts). WFCAM, for example, does use multiple DA computers but the 4 pipelines work independently.

In order to support multiple flag files for a single observation, the detection loop has been modified to allow a frame class to return the names of multiple flag files. When all the flag files are present the pipeline will read the contents and collate a master list of all the raw data files that should be processed. This all assumes that flag files are synchronized between DA computers and are written as an atomic operation (so the pipeline does not do a partial read as the flag file contents are being written).

One very different behaviour for the SCUBA-2 pipeline is that the pipeline will now handle incremental updates to the flag file. This is vital if the summit pipeline is to keep as close as possible to the data acquisition. The ORAC-DR detection loop now reads the current flag file and compares it with a list of files that have already been processed, any new files are read and a new Frame object is sent to the pipeline. This means that the pipeline will be no further behind than a single file (rather than having to wait for an observation to complete).

**Req. PR16**

This effectively means that the QL and science pipelines receive data at the same rate in SCAN mode. Only in DREAM mode do the detection rates differ since reconstructed images appear on the DRAMA parameter as soon as they are created (every few seconds) whereas complete files are only written to disk every minute or so.

As with the standard flag file detection, the incremental read of a flag file can only work if the writes to the flag file are atomic operations. Also, the pipeline will not be able to dynamically detect a variable number of sub-arrays. Once the pipeline has been configured for 4 sub-arrays it will need to be stopped and reconfigured to be able to work with, say, only 3 sub-arrays.

## 5.3 Observation and sub-scan numbers

In initial release ORAC-DR will depend solely on the flag files for data detection. This means that there will be no need to extend data detection to support sub-scans (since flag files do



not include them) but does mean that flag files must be retained even when processing the data off-line and the `-from` and `-list` options will not work with SCUBA-2.

## 5.4 Header Translation

One consequence of multiple files created by multiple DA systems is that the FITS headers for each file are going to be extremely similar, varying in only a few headers. Rather than retaining a FITS header for each input data segment (there may be as many as 60 processed DREAM images per minute per sub-array, corresponding to 244 FITS headers  $((60+1)*4)$  per minute), ORAC-DR now analyzes the headers to create a primary frame header with the static values and a difference sub-header for each component data segment (which may correspond to a processed DREAM quadrant or a single scan from a single sub-array).

Additionally, all FITS information used by the pipeline recipes is translated to a standard internal form (see the `Astro::FITS::HdrTrans` perl module) to allow interoperability of some primitives and recipes for different instruments than originally intended.

## 5.5 Recipe switching

**Req. OR1**

**Req. OR2**

Requirement **OR1** implies that the Pipeline should be able start processing either from the DA-system processed data products (e.g. from DREAM or STARE mode images) or from the raw data frames themselves, or that the Pipeline should be allowed to spend more time processing the data off-line than on-line. Since the DR Recipes specified in the FITS headers will in general have the same final output product (e.g. the `BRIGHT_POINT_SOURCE` Recipe will still be calculating the flux density of the source and optimizing processing for a bright point source), there will be a switch added to ORAC-DR to allow different Recipes to be picked up when the Pipeline is run off-line. For example `BRIGHT_POINT_SOURCE` could be the on-line Recipe name and `BRIGHT_POINT_SOURCE_CAREFUL` could be the off-line Recipe. The only difference is that for the off-line case ORAC-DR would be run as

```
oracdr -suffix _CAREFUL
oracdr -suffix _QL
```

The `-suffix` option would indicate that whenever a Recipe is to be read from the disk, preference should be given to a Recipe that contains the specified suffix. Multiple suffices can be specified using a comma-separated list, such that the first suffix in the list would be chosen in preference to later entries in the list.

This parameter could also be read from a configuration file (see next section).

## 5.6 Recipe parameters

One of the main constraints in ORAC-DR is the inability to adjust the behaviour of a recipe without explicitly editing the recipe itself. This is a barrier to entry for some users and reduces flexibility. For the SCUBA-2 pipeline ORAC-DR will be extended to support a configuration file (by default named `.oracdr` and picked up either from `ORAC_DATA_OUT` or the



user's home directory) that will allow per-primitive and per-recipe configuration. This configuration file will use the Apache format; a simple format based on keyword-value syntax with the added advantage of being hierarchical:

```
# Preferentially choose Quick Look recipes
recipe_suffix  _QL

# Run aperture photometry with 30 arcsec radius
<Primitive _FIND_INTEGRATED_INTENSITY_ >
  Aperture  30
</Primitive>
```

In conjunction with this change, a new global recipe parameter, `$Cfg` will be available to the programmer. This will be a configuration object and will allow the primitive writer to query the configuration and also set a configuration (potentially simplifying the use of steering primitives which currently use the user-defined header system). All primitive parameters (whether from the recipe or from the config file) will be available through the configuration object and the old hash-based primitive argument scheme will be deprecated for new code.

## References

- [1] A. Allan, T. Jenness, F. Economou, M. J. Currie, and M. J. Bly. Generic Data Pipelining Using ORAC-DR. In *ASP Conf. Ser. 281: Astronomical Data Analysis Software and Systems XI*, pages 311–+, 2002.
- [2] J. A. Bailey, T. Farrell, and K. Shortridge. DRAMA: an environment for distributed instrumentation software. In *Proc. SPIE Vol. 2479, p. 62-68, Telescope Control Systems, Patrick T. Wallace; Ed.*, pages 62–68, June 1995.
- [3] B. Cavanagh, P. Hirst, T. Jenness, F. Economou, M. J. Currie, S. Todd, and S. D. Ryder. ORAC-DR: One Pipeline for Multiple Telescopes. In *ASP Conf. Ser. 295: Astronomical Data Analysis Software and Systems XII*, pages 237–+, 2003.
- [4] A. J. Chipperfield. ADAM. Starlink User Note 144, Starlink Project, CCLRC, 2001.
- [5] M. Currie, G. Wright, A. Bridger, and F. Economou. Data Reduction of Jittered Infrared Images Using the ORAC Pipeline. In *ASP Conf. Ser. 172: Astronomical Data Analysis Software and Systems VIII*, pages 175–+, 1999.
- [6] Malcolm J. Currie and Brad Cavanagh. ORAC-DR – imaging data reduction. Starlink User Note 232, Starlink Project, CCLRC, 2003.
- [7] F. Economou, A. Bridger, G. S. Wright, N. P. Rees, and T. Jenness. The Future of Data Reduction at UKIRT. In *ASP Conf. Ser. 145: Astronomical Data Analysis Software and Systems VII*, pages 196–+, 1998.
- [8] Frossie Economou, Tim Jenness, Malcolm J. Currie, Andy Adamson, Alasdair Allan, and Brad Cavanagh. ORAC-DR – overview and general introduction. Starlink User Note 230, Starlink Project, CCLRC, 2002.
- [9] D. Giaretta, M. Taylor, P. Draper, N. Gray, and B. McIlwrath. HDX Data Model: FITS, NDF and XML Implementation. In *ASP Conf. Ser. 295: Astronomical Data Analysis Software and Systems XII*, pages 221–+, 2003.



- [10] Norman Gray. Hdx overview. URL: <http://www.astro.gla.ac.uk/users/norman/note/2003/hdx-overview/>, 2003.
- [11] Paul Hirst and Brad Cavanagh. ORAC-DR – spectroscopy data reduction. Starlink User Note 236, Starlink Project, CCLRC, 2003.
- [12] G. J. Hovey, T. A. Burgess, R. V. Casorso, W. R. Dent, P. E. Dewdney, B. Force, J. F. Lightfoot, A. G. Willis, and K. K. Yeung. New spectral line multibeam correlator system for the James Clerk Maxwell Telescope. In *Proc. SPIE Vol. 4015, p. 114-125, Radio Telescopes, Harvey R. Butcher; Ed.*, pages 114–125, July 2000.
- [13] T. Jenness and F. Economou. The SCUBA Data Reduction Pipeline: ORAC-DR at the JCMT. In *ASP Conf. Ser. 172: Astronomical Data Analysis Software and Systems VIII*, pages 171–+, 1999.
- [14] Tim Jenness. SCUBA-2 DA/DR interface control document. SCUBA-2 Project SC2/SOF/IC210/01, 2003.
- [15] Tim Jenness. SCUBA-2 data reduction pipeline algorithm engine interface. SCUBA-2 Project SC2/SOF/IC210/03, 2003.
- [16] Tim Jenness. SCUBA-2 data reduction recipes and primitives. SCUBA-2 Project SC2/SOF/S210/004, 2003.
- [17] Tim Jenness and Simon Cozens. *Extending and Embedding Perl*. Manning Publications, Inc., 2002. ISBN 1-930110-82-0.
- [18] Tim Jenness and Frossie Economou. The ACSIS-DR/ORAC-DR interface. JCMT OCS/ICD/011, 2001.
- [19] Tim Jenness and Frossie Economou. ORAC-DR – programmer’s guide. Starlink User Note 233, Starlink Project, CCLRC, 2001.
- [20] Tim Jenness and Frossie Economou. ORAC-DR – SCUBA pipeline data reduction. Starlink User Note 233, Starlink Project, CCLRC, 2001.
- [21] Tim Jenness, John F. Lightfoot, Wayne S. Holland, Jane S. Greaves, and Frossie Economou. SCUBA Observing Techniques and Data Reduction Pipeline. In *ASP Conf. Ser. 217: Imaging at Radio through Submillimeter Wavelengths*, pages 205–+, 2000.
- [22] B. D. Kelly and A. J. Chipperfield. AMS – the UNIX ADAM message system. Starlink User Note 241, Starlink Project, CCLRC, 2001.
- [23] M. D. Lawden and K. F. Hartley. ADAM – the starlink software environment. Starlink Guide 4, Starlink Project, CCLRC, 1992.
- [24] A. Lawrence, N. Hambly, B. Mann, M. Irwin, R. G. McMahon, J. R. Lewis, and A. J. Adamson. The WFCAM/UKIDSS data archive: problems and opportunities. In *Survey and Other Telescope Technologies and Discoveries. Edited by Tyson, J. Anthony; Wolff, Sidney. Proceedings of the SPIE, Volume 4836, pp. 418-425 (2002).*, pages 418–425, December 2002.
- [25] Jim Lewis, Simon Hodgkin, Dafydd Evans, and Mike Irwin. WFCAM Pipeline Design. Technical report, Cambridge Astronomical Survey Unit, 2003.
- [26] Nick P. Rees. Proposal for SCUBA-2 pipeline infrastructure. SCUBA-2 Project SC2/SOF/S200/011, 2003.
- [27] Stephen Todd. ORAC-DR – integral field spectroscopy. Starlink User Note 246, Starlink Project, CCLRC, 2003.
- [28] Rodney F. Warren-Smith. NDF – Routines for accessing the extensible N-Dimensional Data Format. Starlink User Note 33, Starlink Project, CCLRC, 1995.

