SUN/230.6

Starlink Project
Starlink User Note 230.6

Frossie Economou, Tim Jenness,
Malcolm Currie, Andy Adamson, Alasdair Allan, Brad Cavanagh
Joint Astronomy Centre, Hilo, Hawaii

June 2004

ORAC-DR: Overview and General Introduction
4.1-0

Abstract

ORAC-DR is a general purpose automatic data reduction pipeline environment. It currently supports data reduction for the United Kingdom Infrared Telescope (UKIRT) instruments UFTI, IRCAM, UIST and CGS4, for the James Clerk Maxwell Telescope (JCMT) instrument SCUBA, for the William Herschel Telescope (WHT) instrument INGRID, for the European Southern Observatory (ESO) instrument ISAAC and for the Anglo-Australian Telescope (AAT) instrument IRIS-2. This document describes the general pipeline environment. For specific information on how to reduce the data for a particular instrument, please consult the appropriate ORAC-DR instrument guide.

Contents

1 Introduction to ORAC-DR

An ORAC-DR HowTo.

Description

This document gives a general introduction to the pipeline, what it does and what it will not do. For information on instrument specific functions, see SUN/231 for SCUBA; SUN/232 for IRCAM, UFTI, Michelle, UIST, INGRID, ISAAC and IRIS-2 imaging; SUN/236 for CGS4, Michelle, UIST and IRIS-2 spectroscopy; and SUN/246 for UIST IFU imaging spectroscopy.

What it is

The ORAC-DR data reduction system is intended to be a pipeline reducer for incoming data. It is in use for online data reduction at UKIRT and JCMT for a variety of instruments. There are a number of differences between the ORAC-DR method of reduction and other systems currently in use and observers should not expect ORAC-DR to behave or be used in the same way as those systems.

Firstly, ORAC-DR aims to reduce data to a point where its quality can be assessed; it will not generally produce publication-quality results (though in certain circumstances it may do). Secondly, although ORAC-DR also works offline, it is expected that observers will use their own preferred data reduction package if they wish to work interactively work on their data. The rest of this document summarizes and accounts for the operational differences between the pipeline and existing packages.

It pipes, therefore it is

This is crucial. Everything else about the package is clear once this is grasped. This is not a reduction package like CGS4DR; it is a reduction black box which knows the incoming data types (by their headers) and transparently applies a reduction recipe to them. There is nothing preventing you from running three simultaneous instances of the pipeline, for example to (i) reduce the incoming data in real time, (ii) re-reduce a previous group of files using a different reduction recipe and (iii) reduce and file a single previous observation as a dark. You do this by running three versions of oracdr, using the command-line switches to alter their behavior (recipe, start and end observation numbers to process, graphics options, etc.). Each instance of the pipeline will go through the required files (existing ones or files just arriving on disk as specified on the command line) and reduce them. Once its remit of reduction is complete, it will exit.

Control

The behavior of ORAC-DR is entirely controlled by the command line options entered at startup. From that point on, the system either takes its reduction recipe instructions from the file headers (this is the default) or uses a hardwired recipe given on the command line itself. The recipe is, in fact, the only allowed parameter on the command line - all the rest are options. Once you start

an instance of oracdr up, there is no further control over it; this is a considerable change from the situation with CGS4DR, for example, where the same package remains up once you start it, and changes are made within the package. Corollary: if the pipeline fails to find a required calibration frame, for example, the only logical thing for it to do is exit cleanly, telling you why it did so. There is no control from within the pipeline. Note that once the full ORAC system is available, there will be plenty of pre-checking that your calibration frames will indeed exist and be appropriate. Until that point, the behavior of the system is logical if you understand the underlying philosophy and do not expect to be able to control the pipeline in real time.

Document info

2 ORAC-DR

Jargon

Description

This document describes some of the terms commonly used in the ORAC-DR documentation.

What it says

ORAC-DR is a data-driven data pipeline that uses recipes and primitives to drive external algorithm engines to perform actual data reduction.

What it means

pipeline

Software that runs without external intervention, taking raw data on one end and producing reduced data in the other.

data-driven

Software that does nothing until data arrives, and then looks to the data and not the user for data reduction information.

recipe

A file containing a list of individual data reduction steps.

primitive

A file containing code performing a meaningful data reduction step

engine

An external package (e.g. a Starlink task) that performs a certain algorithmic step (e.g. addition).

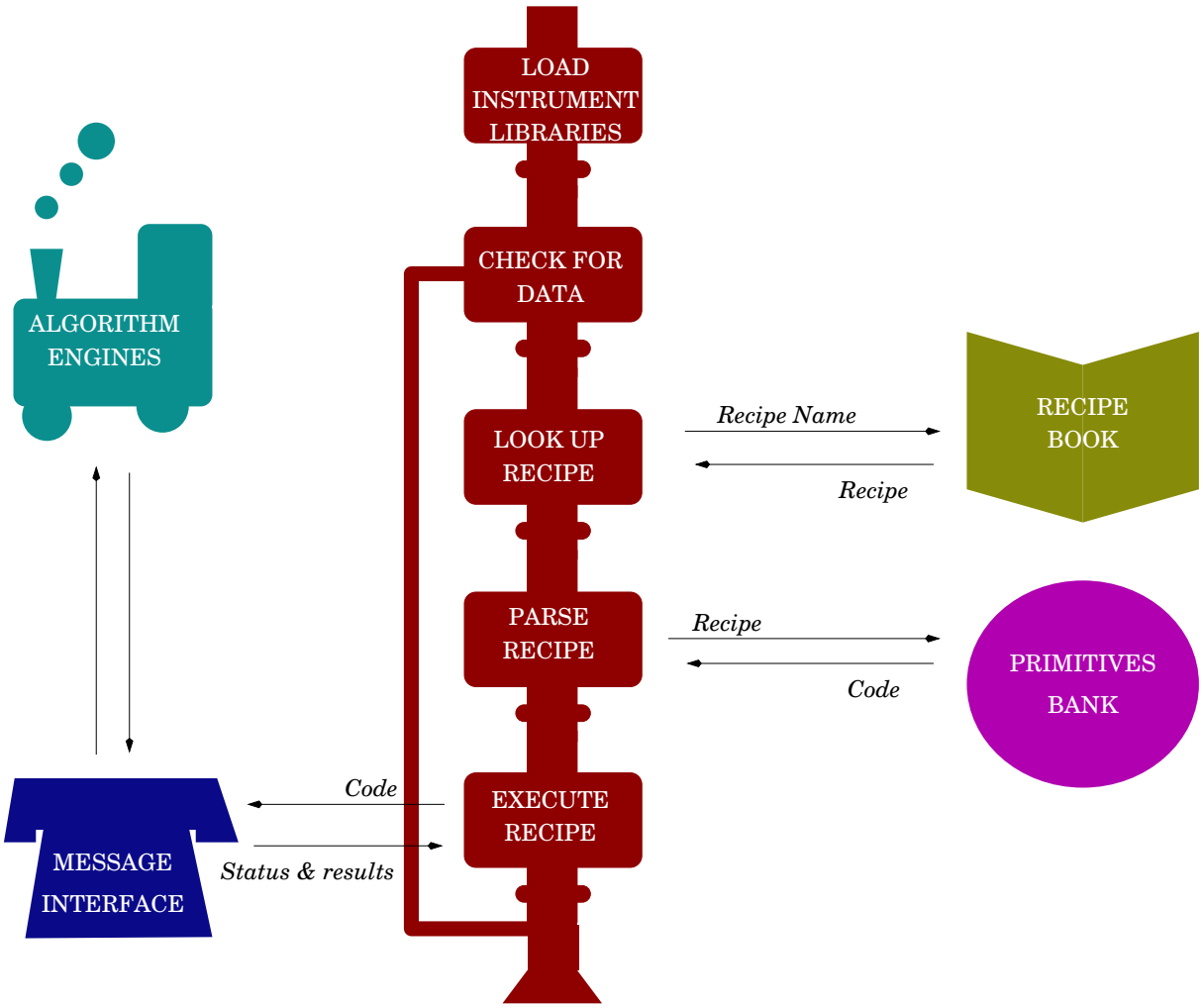

Figure 1: What happens in ORAC-DR

3 Setting up to run oracdr

An ORAC-DR HOWTO

Description

This document describes how to set up the ORAC-DR software for your environment.

The short way (Starlink and JAC users only)

- If your data conforms to the directory naming convention of the instrument, type:

  ```
  setenv ORAC_DATA_ROOT <root data directory>
  oracdr_<instrument> YYYYMMDD
  ```

  You can set up this variable by hand or in your own login script **before** running the oracdr instrument setup script.

  For example, the naming convention for UFTI data is `ufti_data/YYYYMMDD/raw/` for the location of raw data and `ufti_data/YYYYMMDD/reduced/` for the location of reduced data. You can set `ORAC_DATA_ROOT` to the directory in which the UT directory is found. So if your raw UFTI data is in `/home/user/data/UKIRT/ufti_data/YYYYMMDD/raw/` you should type:

  ```
  setenv ORAC_DATA_ROOT /home/user/data/UKIRT/
  oracdr_ufti
  oracdr [-options] [RECIPE]
  ```

  in order to reduce your UFTI data.

- If your raw and reduced data are in arbitrary directories, type:

  ```
  oracdr_instrument <YYYYMMDD>
  setenv ORAC_DATA_IN <raw data directory>
  setenv ORAC_DATA_OUT <reduced data directory>
  ```

  e.g.

  ```
  oracdr_ufti 19990602
  setenv ORAC_DATA_IN /home/user/data/patt99/raw/
  setenv ORAC_DATA_OUT /scratch/user
  oracdr [-options] [RECIPE]
  ```

Note that ORAC-DR works exclusively in `ORAC_DATA_OUT`, irrespective of what your current directory is when you invoke it.

The long way

ORAC-DR uses a number of environment variables for configuration. If you are using a non-Starlink non-JAC installation of ORAC-DR, please consult *ShellVariables* (see appendix) for the complete set of variables and their meaning.

Document info

Original author: frossie

4 ORAC-DR Components

An ORAC-DR HowTo

Description

This document describes the executables that form part of the ORAC-DR distribution.

Set up scripts

These are the commands you have to type to set up ORAC-DR for the various supported instruments.

`oracdr_cgs4`

Sets up for the data reduction of CGS4.

`oracdr_ircam`

Sets up for the data reduction of IRCAM, the UKIRT Infrared Camera.

`oracdr_michelle`

Sets up for the data reduction of MICHELLE.

`oracdr_scuba`

Sets up for the data reduction of SCUBA, the JCMT Submillimetre Common User Bolometer Array.

`oracdr_ufti`

Sets up for the data reduction of UFTI (the UKIRT Fast Track Imager)

`oracdr_iris2`

Sets up for the data reduction of IRIS-2 (the AAT Infrared Imaging Spectrograph)

`oracdr_uist`

Sets up the data-reduction system for UIST (The UKIRT infrared imaging spectrometer with IFU).

`oracdr_ingrid`

Sets up the data reduction system for INGRID.

`oracdr_isaac`

Sets up the data-reduction system for ESO ISAAC in imaging modes. Support for spectroscopy modes is ongoing.

`xoracdr`

Initialises the environment and then launches the ORAC-DR GUI.

Executables

`oracdr`

Runs the pipeline. For more information see *oracdr*

`oracman`

Displays man-page information for all ORAC-DR executables, recipes, primitives and how-to guides. Type:

```
oracman <foo>
```

for documentation, e.g.

```
oracman oracdr
oracman ARRAY_TESTS
oracman _SUBTRACT_DARK_
oracman DisplaySystem
```

`oracdisp`

Sets up the X display configuration tool. For more information on `oracdisp` and the display system see *oracdisp* and *DisplaySystem*

`oracdr_nuke`

Kills `oracdr`, any related processes (including any Starlink processes), and clears shared memory owned by the user. Useful if something happens to cause ORAC-DR to hang. See oracdr_nuke

Document info

5 Xoracdr

An ORAC-DR HOWTO

DESCRIPTION

The ORAC-DR pipeline can be controlled from the Xoracdr application. This note describes the user interface of Xoracdr and how to use it.

STARTING XORACDR

The ORAC-DR pipeline and Xoracdr software uses a number of environmental variables for configuration. For non-Starlink, non-JAC installation of ORAC-DR, please consult *ShellVariables* (see appendix) for more information.

To start the Xoracdr application type `xoracdr` at the prompt:

```
% xoracdr
ORAC Data Reduction Pipeline -- (ORAC-DR Version 3.0)

Please wait, spawning Xoracdr...
```

this will run the Xoracdr setup script and bring up the Xoracdr main window.

OPTIONS AND ARGUMENTS

Unlike command line control of the ORAC-DR pipeline, configuration of the pipeline options is via the main window. There are therefore only a few command line options

-vers

Displays revision information about Xoracdr and the version of Perl it is using.

-ut *YYYYMMDD*

UT date of observations (defaults to current YYYYMMDD).The UT is required for UKIRT and JCMT instruments as it forms part of the file naming convention for data files.

-honour

If you start xoracdr with the `ORAC_INSTRUMENT` environment variable unset, i.e. you have not run one of the instrument setup scripts, but have the `ORAC_DATA_OUT` environment variable set you may wish to force Xoracdr to honour this using the `-honour` command line option.

USING XORACDR

If your data conforms to the directory naming convention for the instrument, setting up the pipeline is fairly trivial.

Select your instrument from the list box to the left hand side of the application main window and then enter the UT date of the observations you are reducing into the entry field in the center of the main window. The raw and reduced data paths should now be set correctly. Enter the range of frame numbers that you are interested in into the `List` text entry and push the `Start ORAC-DR` button. The pipeline should initalise and start processing you data immediately.

A more detailed description of the Xoracdr user interface follows for those people who need to do something slightly out of the ordinary, although the interface itself should be fairly self explanatory for those people used to the *oracdr* command line interface.

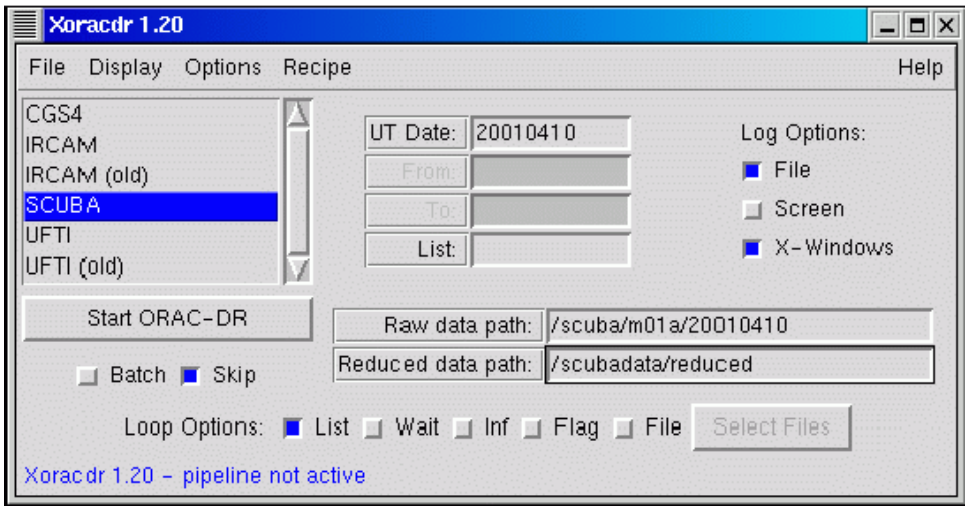

Figure 2: The Xoracdr main window

THE MAIN WINDOW

The Xoracdr main window is divided into three main sections. At the top is the menu bar with `File`, `Display`, `Options`, `Recipe` and `Help` menus.

File Menu

Stop Processing

When the pipeline is running further data processing can be halted by selecting this menu entry. Control will be returned to the main window, processing can be restarted from the beginning by depressing the *Start ORAC-DR* button.

Pause Processing

When the pipeline is running the processing of further data can be paused by selecting this menu item, a pop-up window will appear with a *Resume* button. Processing can be re-started by depressing the *Resume* button in the pop-up.

Nuke ORAC-DR

Selecting this menu option will kill all ORAC-DR related processes, including spawned Starlink monoliths, and will free up any shared memory used by these processes. This will also kill the Xoracdr graphical user interface.

Display Menu

No Display

Selecting this checkbox will instruct Xoracdr not to launch the display system. No data will be displayed and GWM (or GAIA) windows will not be launched.

Configure Display

Allows configuration of the ORAC-DR display environment. Selecting this menu item will display a popup window which can be used to edit the current environment and add display directives to the current environment. This popup can also be run as a stand alone application from the command line, e.g.

```
% oracdisp
```

for more information on how to use the display environment editor see the *oracdisp* documentation and the documentation on the *DisplaySystem*.

Options Menu

Allow Resume

Allow the pipeline to resume midway through the processing of a group so long as the recipe/instrument supports this behaviour. Default is for the group file to be deleted when a new group is created. When this menu option is selected, the group file is retained. **NOTE** this option is not currently supported by IRCAM, UFTI and SCUBA recipes.

No Engines

Do not start algorithm engines. **NOTE** this will cause the vast majority of recipes to fail.

Common ADAM

Do not create an invocation specific temporary directory for the messaging systems but use whatever directory is the usual default. For ADAM tasks this would mean that `~/adam` or `$ADAM_USER` will be used rather than a private ORAC-DR directory. This should only be used when it is required for ORAC-DR to talk to tasks that were not started by the pipeline and could lead to possible confusion if multiple pipelines are running using this flag.

Verbose

If selected then messages from the Starlink engines will be printed in addition to the normal ORAC-DR messages

Debug

Log debug messages to file `ORACDR.DEBUG` in `$ORAC_DATA_OUT`.

Warn

Turn on perl level warning messages (`perl -w`). This should be used for debugging only. If `-verbose` is also true then full perl diagnostics are turned on (see *diagnostics* for more information on this perl pragma).

Beep

Is selected the pipeline will make as much noise as possible over errors and pipeline exit.

Setup Environment

If selected a window will appear allowing you to enter file paths for the data and calibration root directories, user recipe and primitive directories, the raw and reduced data directories and the instrument calibration directory. If these have already been set, either by selecting an instrument from the listbox or from environment variables you can override the default options using this popup window. **NOTE** The raw and reduced data paths will be overridden if you select an instrument from the main window listbox after entering values into this window.

Calibration Options

This item will remain disabled until an instrument is selected in the main window listbox, after this you are free to select this item from the menu. On selection a

window will popup allowing you to enter calibration options for the instrument, see the user guide for your instrument for more information.

Recipe Menu

Show Current Recipe

Selecting this menu item will ensure that the recipe display window will appear along with the log window. The recipe display window will show the currently executing recipe and a list of the primitives contained in the recipe. The currently running primitive will be highlighted.

Override: *RECIPE*

This item will be greyed out until an override recipe is selected using the *Override Recipe* option further down the `Recipe` menu. After an override recipe is selected then this option will become active. If the checkbox is selected then the selected recipe will override the one specified in the file headers. This override recipe is used for all data files regardless of header contents or observation mode, so make sure you only only apply it to appropriate data frames.

Set Override Recipe

This item will remain disabled until an instrument is selected in the main window listbox, after which if selected it will allow you to choose an *Override Recipe*

Edit Recipe

Allows you to select and then edit a recipe. The recipe will be saved to `ORAC_RECIPE_DIR`, the user recipe directory, which if unset will default to `ORAC_DATA_OUT`. Recipes in `ORAC_RECIPE_DIR` will be run in preference to those in the instrument recipe directories of the same name.

Help Menu

Selecting an entry from the `Help` menu will popup a help browser to display the relevant documentation. At the bottom on the menu there is also an `About XORAC-DR` entry which will display the licence terms and conditions for the ORAC-DR software.

At the bottom of the main window is the status bar, this reports the status of the pipeline process. It will display the currently running recipe name, along with other status information.

Between the menu and status bar is the bulk of the main window. On the left is the instrument list box, this allows you to select the instrument whose data you wish to process. If the `ORAC_INSTRUMENT` environment variable is set before starting Xoracdr the instrument will be preselected on startup.

Once an instrument is selected the user interface will be configured into a default state for the instrument. **NOTE** it is important to select an instrument before doing any customization of the settings as your changes may be overridden by the default options imposed by setting up an instrument.

At this point the *Start ORAC-DR* button will become enabled and you can start the pipeline if the configuration options are to your liking.

FURTHER CONFIGURATION

If not already set using the `-ut` command line option, and if you are using a loop type where it is necessary to build the filename using the UT date (i.e. all loop options except *file*) the UT date of the observation run can be set via the relevant text entry box in the middle of the main window. **NOTE** Doing so will run the instrument setup routine again which will override some environment options such as `ORAC_DATA_IN` and `ORAC_DATA_OUT`.

If ORAC-DR is being run in post-observation mode, the default data detection loop is *list*. Other loop options can be selected using the series of checkboxes along the bottom of the main window. The available options are

list

The pipeline will stop once the observations in the list have been reduced.

wait

Waits for data to appear before timing out. Data is reduced and the pipeline waits for the next file.

inf

Do not wait for data. Simply reduce data starting with observation specified by `from` and continuing until no more files are present. This is the fastest way of reducing data offline.

flag

Waits for completion files to appear (flags) before processing the data. Data is reduced and the pipeline waits for more data by checking the presence of a flag.

file

Works much like the `list` loop option except that looping is carried out over a list of arbitrarily named files. When the `file` loop option is selected the *Select Files* button will be enabled allowing you to generate this list. **NOTE** As well as having arbitrary filenames, files can be added in arbitrary order, this allows you to reduces (for instance) all the calibration frames for a night first, followed by the actual observations.

There are, in addition, two other options which affect the looping scheme, these being `batch` and `skip`.

batch

Run in batch mode. Precalculate groups before processing data. 'wait' loop mode should not be used with this option. **NOTE** only SCUBA recipes support this option.

skip

Allow the data detection loop to skip missing observations. Default is to stop the loop when an expected file can not be found.

See the *DataLoops* documentation for more information on looping schemes.

LOG OPTIONS

Finally, you can configure the log options from the three checkboxes located to the right hand side of the main window. By default logging is sent to a file and to an Xwindow.

SEE ALSO

oracdr, *oracdisp*, oracdr_parse_recipe

AUTHOR

Alasdair Allan (aa@astro.ex.ac.uk)

COPYRIGHT

6 oracdr

ORAC Data Reduction pipeline

Synopsis

```
oracdr [-options] [RECIPE]
oracdr -from 5
oracdr -ut 19990523 -list 15:35,40,44 -batch
```

DESCRIPTION

`oracdr` is the actual ORAC-DR data reduction pipeline. This document describes the command line options that can be used to modify the pipeline operation.

Arguments

The following argument is supported:

- RECIPE

  By default, ORAC-DR looks in the file header for the name of the recipe to be used on the data. If you specify the name of a recipe on the command line, it will override the one specified in the header. This override recipe is used for all data files regardless of header contents or observation mode, so make sure you only only apply it to appropriate data frames.

Options

All ORAC-DR behaviour is controlled by the option switches. These options may be abbreviated to a unique substring. It is via command line switches that you (for example) control the range of file numbers to be reduced, force the system to use a particular calibration file when reducing (e.g. to try a different flat exposure). This list needs to be read thoroughly by anyone wanting to use the system.

General Options

-help

List help text. This prints a summary of this document.

-version

Print the version number.

-verbose

Print messages from the Starlink engines (rather than just ORAC-DR messages).

-man

Print the full documentation.

-debug

Log debug messages to file `ORACDR.DEBUG` in `$ORAC_DATA_OUT`.

-warn

Turn on perl level warning messages (`perl -w`). This should be used for debugging only. If `-verbose` is also true then full perl diagnostics are turned on (see *diagnostics* for more information on this perl pragma).

-beep

Make as much noise as possible over errors and pipeline exit. Default is not to beep.

-rechelp

Show help text for the given recipe.

Windows and output

-nodisplay

Do not launch the display system. No data will be displayed and GWM, GAIA etc. windows will not be launched.

-showcurrent

Launch a recipe viewer window along with the log Xwindow

-log s

Log to terminal screen (standard out)

-log f

Log to a file. The logfile is called `.oracdr_NNNN.log` where NNNN is the current process ID. It is written to `$ORAC_DATA_OUT` and is a hidden file.

-log h

Log to a file using HTML to provide formatting. The logfile is called `.oracdr_NNNN.html` where NNNN is the current process ID. It is written to `$ORAC_DATA_OUT` and is a hidden file.

-log x

Log to an X window. Has the advantage that warnings, errors and results are written to different, independently scrollable windows. The plus and minus keys can be used to adjust the font size.

The three log options can be combined. The default is `-log sx`

To run ORAC-DR using output only within the xterm that you used to invoke it in, use `-nodisplay -log s`. This is the fastest way to run the pipeline if you are not interested in visually inspecting the data as it is being reduced.

Observations

-from

Number of first observation.

-to

Number of last observation.

-list

Comma separated list of observation *numbers*. Colons indicate a range. For example, '1,2,4:6,10' means 1,2,4,5,6,10.

-files

File name of a flat ASCII text file containing a list of observation *files* to be reduced, one file per line. Path information should be either relative to `$ORAC_DATA_IN`, or the absolute path.

UT date

-ut

UT date of observations (defaults to current yyyymmdd). When the instrument specific setup scripts are run, oracdr is automatically aliased to use the correct `-ut` option. The UT is required for UKIRT and JCMT instruments as it forms part of the file naming convention for data files.

Recipe Selection and Modification

-recsuffix

Modify the recipe search algorithm such that a recipe variant can be selected if available. For example with '`-recsuffix QL`' a recipe named MYRECIPE_QL would be picked up in preference to MYRECIPE.

Multiple suffices can be supplied using a comma separator.

```
-recsuffix QL1,QL2
```

-recpars

Recipe behaviour can be controlled by specifying a recipe parameters file. This is a file in INI format with a block per recipe name.

```
[RECIPE_NAME]
param1=value1
param2=value2
```

Parameters to be supplied for all recipes can also be specified explicity, overriding any parameters in the file. Note however that only one file may be specified. For example:

```
-recpars="tilenum=456356,recpars-M95AC34.ini"
```

-headeroverride

Specify a file containing header information which should be used to override that found in the data files. Typically used only for legacy data where headers are missing or incorrect. The file is in INI format, with one block per file, e.g.:

```
[x19990401_00009]
DRRECIPE=REDUCE_DARK
GRPNUM=9
GRPMEM=T
NOFFSETS=6

[x19990401_00010]
DRRECIPE=JITTER_SELF_FLAT
GRPNUM=10
GRPMEM=T
NOFFSETS=41
```

--preprocess

Enables "WESLEY" pre-processing mode.

Calibration options.

-calib

Used to specify calibration overrides. Accepts comma separated key=value pairs. (e.g. '`-cal dark=file1`' or '`-cal dark=file1,bias=file2`'). The allowed options depends on the instrument that is in use.

See *Calibrating* for more information on how the pipeline deals with calibrations.

Looping options

The `-loop` option specifies the type of data detection loop. Allowed values are 'list', 'inf', 'wait', 'flag' or 'file'. In almost all cases of offline use, 'inf' is most appropriate.

-loop list

Default when using the `-list` option. The pipeline will stop once the observations in the list have been reduced.

-loop wait

Waits for data to appear before timing out. Data is reduced and the pipeline waits for the next file.

-loop inf

Do not wait for data. Simply reduce data starting with observation specified by `-from` and continuing until no more files are present. Implicitly used when `-from` is specified. This is the fastest way of reducing data offline.

-loop flag

Waits for completion files to appear (flags) before processing the data. Data is reduced and the pipeline waits for more data by checking the presence of a flag.

-loop file

Works much like `-loop list` except that looping is carried out over a list of arbitarly named files input from the `-files` command line option.

-loop task

Obtain data from a remote (DRAMA) task.

-flagsync

Read flag files "in sync". I.e., if there are multiple flag files for an observation, as is the case for SCUBA-2, only read matching numbers of entries from these files. Entries in any flag files in excess of the minimum number will not be read, instead being left for potential future processing.

See *DataLoops* for more information on looping schemes.

Group processing options

-batch

Run in batch mode. Precalculate groups before processing data. 'wait' loop mode should not be used with this option. **NOTE** only JCMT recipes support this option.

-skip

Allow the data detection loop to skip missing observations. Default is to stop the loop when an expected file can not be found.

-skiperror

Continue after errors thrown in processing. This setting should not be used at the telescope.

-resume

Allow the pipeline to resume midway through the processing of a group. (so long as the recipe/instrument supports this behaviour). Default is for the group file to be deleted when a new group is created. When `-resume` is set, the group file is retained. **NOTE** this option is not currently supported by IRCAM, UFTI and SCUBA recipes.

-grptrans

Groups are presumed to be transinet and no longer needed when a new group is created. This is useful when you know that groups can not be broken up. Has no effect in batch mode. Memory usage will be significantly lower if many hundreds of frames and groups are to be processed.

This option is not the same as setting the ORAC_NOGROUPS environment variable. That environment variable disables all group processing whereas this command line option ensures that only a single group is being processed.

-onegroup

All given observations and files are processed in the same group. Be careful in using this option, as sometimes this may not be what you want (i.e. if you're processing ACSIS data at two different frequencies).

The Frame grouping string is not affected.

-groupid

Forces the grouping string to take the specified value. This means that all frames will be combined into a single group as for "-onegroup" but the grouping string in each Frame will take this value. This is important if the string should end up in output group files (e.g. the ASN_ID in JCMT Science Archive data).

Engine Options

-noeng

Do not start algorithm engines. **NOTE** this will cause the vast majority of recipes to fail.

-nomsgtmp

Do not create an invocation specific temporary directory for the messaging systems but use whatever directory is the usual default. For ADAM tasks this would mean that ~/adam or $ADAM_USER will be used rather than a private ORAC-DR directory. This should only be used when it is required for ORAC-DR to talk to tasks that were not started by the pipeline and could lead to possible confusion if multiple pipelines are running using this flag.

AUTHORS

Frossie Economou (frossie@jach.hawaii.edu), Tim Jenness (t.jenness@jach.hawaii.edu), Alasdair Allan (aa@astro.ex.ac.uk), Brad Cavanagh (b.cavanagh@jach.hawaii.edu)

COPYRIGHT

7 oracdr_monitor

Monitor the ORAC-DR pipeline output

SYNOPSIS

```
oracdr_monitor

oracdr_monitor --tty

oracdr_monitor --nodisplay
```

To automatically determine ORAC_DATA_OUT based on a location file:

```
oracdr_scuba2_850_summit

orcadr_monitor --useloc
```

DESCRIPTION

This program monitors the output from ORAC-DR log files in the ORAC_DATA_OUT directory. If a new log file is created (e.g. by a restart of ORAC-DR) the program will automatically begin monitoring the new one.

If an X-display is running it sends the output to an Xwindow similar to the standard ORAC-DR logging window. If there is no display the output is sent to standard output.

By default the data display is enabled to monitor the live pipeline display requests. On startup only new display requests are handled (any pending from earlier in the pipeline processing are ignored). The data display can be disabled using the `-nodisplay` option.

OPTIONS

--help

List help text. This prints a summary of this document.

--man

Print the full documentation.

--version

Print the version number.

--tty

This sends the output to the current xterm. It is automatically selected if the DISPLAY environment variable is unset.

--nodisplay

Disable display monitoring. Ignored if DISPLAY environment variable is not set.

--skip

If display monitoring is enabled, by default the monitor will attempt to process every request even if that means it falls behind the pipeline. With this option only the most recent request will be handled with the exception that Group display requests are always processed.

--uselocation

Use the location "declared" by a file in ORAC_LOCATION_DIR (default /jac_sw/oracdr-locations) as ORAC_DATA_OUT.

--recsuffix

Specifies a preferred recipe suffix. This affects the pipeline location file used if the `--uselocation` option is given.

AUTHOR

Tim Jenness <t.jenness@jach.hawaii.edu>

COPYRIGHT

8 ORAC-DR

Credit and License

Description

This document describes the credit and licensing information for ORAC-DR.

Acknowledgements

ORAC-DR was developed at the Joint Astronomy Centre by Frossie Economou and Tim Jenness in collaboration with the UK Astronomy Technology Centre as part of the ORAC project. Other members of the ORAC-DR team have subsequently contributed to ORAC-DR's infrastructure.

Other members of the ORAC team provided invaluable scientific and technical input to ORAC-DR, in particular Alan Bridger (UKATC), Gillian Wright (UKATC) and Andy Adamson (JAC).

Software support for SCUBA was added by Tim Jenness. Malcolm Currie provided the software for UFTI, IRCAM, and Michelle imaging. Support for CGS4 and Michelle spectroscopy was added by Paul Hirst, Frossie Economou, Tim Jenness, Malcolm Currie, and Brad Cavanagh. The ORAC-DR GUI was added by Alasdair Allan (Starlink). IRIS-2 support was added by Brad Cavanagh in conjunction with Stuart Ryder (AAT). ISAAC and INGRID support was added by Malcolm Currie. UIST IFU support was added by Stephen Todd.

The developers are also grateful to numerous members of UKIRT and JCMT staff for their ideas and feedback.

Copyright and License

If you have used ORAC-DR for your data reduction please acknowledge it in your publications. It costs you nothing and gives us a warm fuzzy feeling.

Recommended publications for ORAC-DR itself are:

The Future of Data Reduction at UKIRT, Economou, F.; Bridger, A.; Wright, G. S.; Rees, N. P. and Jenness, T., 1998, ASP Conf. Ser., 145, 196

or

The ORAC-DR data reduction pipeline Cavanagh, B.; Jenness, T.; Economou, F. & Currie, M. J., 2008, Astron. Nach., 329, 295

For the SCUBA pipeline please use: The SCUBA Data Reduction Pipeline: ORAC-DR at the JCMT, Jenness, Tim & Economou, Frossie, 1999, ASP Conf. Ser., 172, 171

For SCUBA-2: Design of the SCUBA-2 Quick Look Display and Data Reduction Pipeline, Gibb, A. G.; Scott, D.; Jenness, T.; Economou, F.; Kelly, B. D. & Holland, W. S., 2005, ASP Conf. Ser., 347, 585

ORAC-DR is registered with the Astrophyiscs Source Code Library as ascl:1310.001

Document info

9 Release Notes

V4.0

- Support for UIST in all observation modes.
- Support for INGRID in all observation modes.
- Support for ISAAC in imaging mode, and preliminary support for spectroscopy mode.
- New document, SUN/246, describing integral field spectroscopy reduction and recipes.
- New `ORAC_KEEP` environment variable to retain intermediate frames.
- Spectroscopy:
  - Widened optimal extraction windows for better profile fitting.
  - Flux calibration for I-band.
- Imaging:
  - Modification of EXTRACTOR object-detection parameters to obtain a flatter, more accurate flat-fielded mosaic.
  - Offset patterns need not be centered at centre of the array.
  - Four new recipes including NOD_SKY_FLAT_THERMAL recipe for reduction of thermal data using sky observations for flat-fielding.
  - REDUCE_DARK supports variance creation and propagation by default.
  - Expanded SUN/232 with more description of the primitives, and information for programmers wishing to adapt the recipes.
- SCUBA:
  - CSO Tau fits up-to-date to January 2003 (when the tau meter broke).

  - Flux Conversion Factors verified up to March 2003.
  - More robust error handling for poor data.

V3.1
- Support for AAT IRIS2 data
- Spectroscopy:
  - Extracts "sky-arcs" to enable wavelength calibration of Michelle data.
  - Now handles offset patterns that don't originate at (0,0).
  - Peak-up routines for Michelle.
  - Single beam polarimetry now much more robust.
  - Masking of off-slit areas of image improved.
  - Better bad pixel detection in flat fields.
- Imaging:
  - Addition of NOD_CHOP_FAINT (faint mid-IR) and NOD_CHOP_SCAN (scan pattern mid-IR) recipes.
  - Addition of ADDWCS (adds WCS to headers) recipe.

V3.0
- Support for Michelle data.
- Support for multi-mode instruments, such as Michelle and UIST.
- Three Michelle recipes for nodded and chopped data (vanilla, photometry and moving target). New NQ standards file.
- Easy to switch on variance creation and propagation; calculates correct data variance for UKIRT IR imagers.
- Faster object masking using EXTRACTOR instead of PISA.
- Better registration of sparse fields using astrometry.
- Tidier output for easier reading. Added content to messages.
- Comments in calibration rules files.
- SCUBA: improvements in calibration of SCUBA data (both for flux conversion factors and extinction correction using `-calib tausys=csofit`).
- Use of internal headers and directory reorganisation, permitting generic-named recipes and primitives, and optimising code use; and easier to add new instruments.

V2.1
- New GUI (xoracdr) to simplify use of the pipeline.
- Enhanced CGS4 and imaging recipes.

V2.0
- Support for CGS4
- SCUBA: Jiggle map calibration
- IR Imaging: Polarimetry support plus recipe generalization.

V1.0
- First release to Starlink. Includes SCUBA and imaging (UFTI FITS and IRCAM) recipes.

A The ORAC-DR Data Loops

An ORAC-DR HOWTO

Description

ORAC-DR may use a variety of ways to detect available data. This document describes what they mean and when to use them.

How the pipeline operates

ORAC-DR is a data-driven pipeline. This means that it does things in response to incoming data and uses the information associated with that data to determine how to process a file. It is also a sequential (i.e. non-parallel) process. This means it only does one thing at a time. As a result, ORAC-DR is always doing one of two things

- Seeks new data
- Reduces data

How the pipeline detects new data

Unless the `-files` option is used, the pipeline starts from looking for observation number 1, unless another number has been specified via the `-from` or `-list` options.

The various `-loop` options determine what the criterion is for concluding that the observation it is waiting for has indeed arrived.

-loop wait

If you use this option, the pipeline monitors the size of the file that is is expecting. For example, if it has just reduced observation number 41, it waits for observation number 42 to appear on disk and watches it size growing as it is being written out by the data handling system. If the file does not grow in size for a certain amount of time it concluded that readout is complete and proceeds with reducing it. Obviously this method is not very robust if the pipeline is operating or network-mounted disks or with acquisition systems that are prone to stalling during readout. However it may be the only option for online data reduction with some data handling systems. This option should be used with IRCAM.

-loop flag

This option instructs the pipeline to monitor not the data file itself, but a "flag" file whose appearance indicates readout completion. Typically this is a zero-length file written by the data acquisition after the data file writing is done. This is most robust in architectures where there is no chance of a data file being written without a flag file or vice versa. This option should be used with SCUBA, UFTI, the WFS and MICHELLE.

-loop inf

Under this option, the pipeline reduces data assuming it is available and keeps going one observation at a time until no more data is to be found (or infinity, whichever comes first!), at which point it terminates. It overrides the `-to` option. This is suitable for offline data reduction of any instrument and is the default option if none is specified.

-loop list

The specified data frames (and/or range of observations) are assumed to be available, are reduced and then the pipeline exits. This option is implied by usage of the `-list` option, or usage of the `-from` and `-to` options in the same invocation. It is unlikely that a user will need to explicitly specify it.

-loop file

The specified data frames are assumed available, are reduced and then the pipeline exists. This option is impled by usage of the `-files` option. Which provides a filename (relative to the current directory) of a flat ASCII text file containing a list of observation *files* to be reduced, one filename per line. The filenames, unlike other loop options used by ORAC-DR, are not based on UT date and may be arbitarily constructed.

B The ORAC-DR Display System

An ORAC-DR HOWTO

Description

The ORAC-DR pipeline has a highly configurable display engine. This note describes how it works and how to use it.

The *disp.dat* file

The actual display engine is configured via an ASCII file called *disp.dat*. A default *disp.dat* file exists in the general oracdr distribution (`ORAC_DIR`). Additionally a default *disp.dat* is provided by your instrument scientist or software engineer in the appropriate calibration directory (`ORAC_DATA_CAL`)

When you invoke oracdr, the pipeline checks in `ORAC_DATA_OUT` for an existing *disp.dat*. If it does not find one, it copies in the one from `ORAC_DATA_CAL`, or if that doesn't exist, the one from `ORAC_DIR`. In all of those scenarios you end up with a file in `ORAC_DATA_OUT` which is what is used by the display system. Any configuration changes have to be reflected in the file.

How to change the *disp.dat* file

As it is an ASCII file, you may edit the file directly in your favourite editor. However a much easier approach is to type use the oracdisp GUI (simply type oracdisp). This has various options available at the top and shows the *disp.dat* entries corresponding to those choices at the bottom. Don't forget to write press the "Configure" button to write the file to disk.

You may use oracdisp (or the editor) to change the display parameters while the pipeline is running.

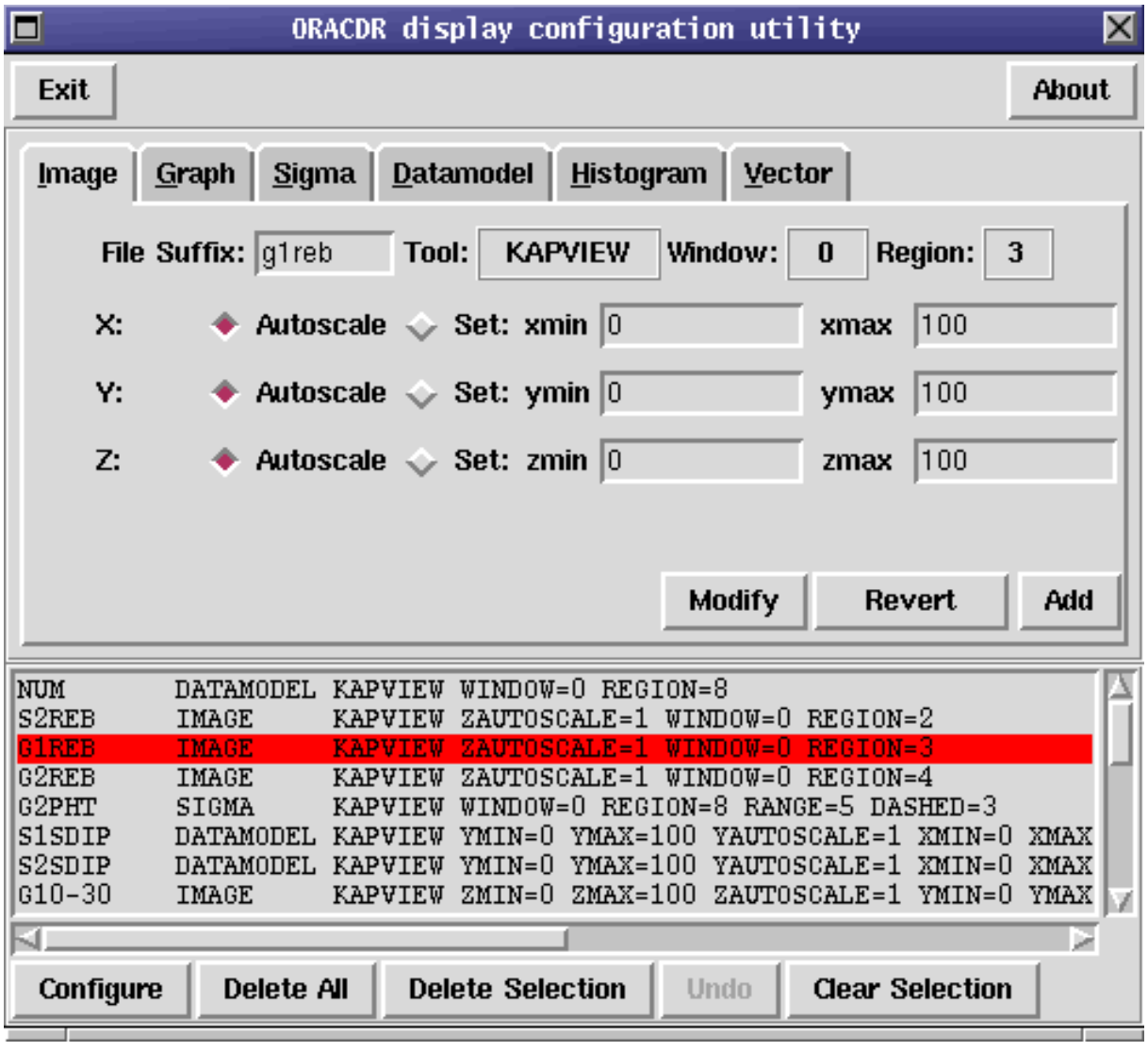

Figure 3: The ORACDISP display configuration tool

How the pipeline displays

The *disp.dat* file has a series of entries consisting of a line each. Each line has a series of space-separated items. All but the first item, the suffix, use the keyword=value syntax.

These items are:

suffix

A file suffix representing a particular step of the data reduction process as designated by that instrument's file naming convention. For example, for UFTI data "dk" states what to do with a file called `f19990330_00042_dk.sdf`. Many entries may be made for a particular suffix. The following special conventions are used: NUM describes a file ending in just an observation number (usually representing raw data). For instruments that have multiple data arrays in a single data frame, S2<suffix> (e.g. S2dk) represents the second data array in the frame with the _dk suffix.

tool

A display tool such as `Gaia` or `KAPVIEW` (the latter being a collective term for various `KAPPA` display tasks). The tools available are determined by the display type (q.v.) selected.

type

A display type such as graph, image, contour etc. These vary according to the tool selected. The following types are available. The tools which support the given display type are listed in parentheses.

contour (`KAPVIEW`)
Plots a contour plot.

datamodel (`KAPVIEW`)
Displays data (as points) with a model overlaid.

graph (`KAPVIEW`, `P4`)
Plots a line graph such as a spectrum.

histogram (`KAPVIEW`)
Plots a line graph such as a spectrum.

image (`GAIA`, `KAPVIEW`, `P4`)
Displays an image.

sigma (`KAPVIEW`)
Draws a scatter plot with a Y range of +/- N standard deviations.

vector (`KAPVIEW`)
Displays an image and vectors (e.g. polarimetry data).

region

For device tools that support it, region addresses the parts of a window where a display ends up. For KAPVIEW, these are: whole screen (0), top left quarter (1), top-right quarter (2), bottom-left quarter (3), bottom-right quarter (4), left half (5), left right (6), top half (7), bottom half (8). Defaults to 0.

window

The number of the window in which the display is to go. The value of this is simply an identifier and does not presuppose order. If you ask for an image display to go to window 2 and you have configured no displays to go to windows 0 and 1 then you will only get one window on your screen. If you then configure a histogram display to go to window 2 it will go to the same window whereas if you configure it to go to window 1 (or 5 or 9 or anything else besides 2) it will end up in its own window. Note that no windows are launched until they are required.

xautoscale

Specifies whether or not to use a section of the data frame. If set to 1, meaning true, the whole X axis is used. If set to 0, the pixel limits are specified by keywords xmin and xmax. The default is 1.

There are corresponding autoscaling keywords for higher dimensions named yautoscale, 3autoscale, 4autoscale etc.

xmin, xmax

The X-axis pixel limits of the data to be displayed when xautoscale=1. xmin should be less than or equal to xmax. There are corresponding pixel-range keywords for higher dimensions: ymin and ymax, 3min and 3max, 4min and 4max etc. when autoscaling on the corresponding axis is disabled.

zautoscale

Set to 1 meaning true, this scales this requests that the data are scaled automatically between the data range. In the case of images on GAIA the cut is at the 95 percentile. When zautoscale is 0, the scaling is between the limits defined by keywords zmin and zmax. Defaults to 1 if absent.

zmin, zmax

When zautoscale is 1, these specifiy the lower and higher scaling limits for the data values.

cut

If the number of dimensions in the data file is greater than that requested, sections in higher dimensions are set to 1 by compressing the undesired dimension(s). Option cut specifies the desired dimension(s).

Option cut is a comma-separated list specifying the dimensionality and axes to retain. The number of entries should equal the dimensionality needed by the type of plot. For instance, a graph only one value is required since a graph is 1-D. The allowed values are X, Y, 3, 4, or 5. If the number of dimensions in the data file is fewer than that requested, ORAC-DR prints a warning message.

Here are two examples, a graph can be displayed from a 2-dimensional image by displaying a cut in the X direction (averaging over the Ys) by setting cut=X. A 'white-light' contour plot of a x,y,wavelength spectral data cube may be plotted using cut=X,Y.

There are also parameters special to particular types of display, which also use the keyword=value syntax. These are:

angrot

The angle to add to all vectors in a type=vector plot.

comp

The array component to display. Allowed values are Data, Variance, or Error. The default is Data. This applies to type=contour, datamodel, graph, histogram, image, or sigma displays.

dashed

The location of the dashed lines for a type=sigma display in standard deviaiton units. This defaults to 3.

errbar

If set to 1 (true), error bars are plotted on a type=graph display, provided there is variance information present. The default is 0, meaning do not plot error bars.

multivector

This controls the appearance of vectors in a type=vector plot. If set to 0 (false), the default, the vectors are white and have thickness 1. If set to 1 (true), the vectors are yellow with a blue trim and have thickness three.

nbins

This is the number of bins to be used for histogram calculation for type=histogram. It defaults to 20.

ncont

This specifies the number of contours to plot for type=contour. It defaults to 6 if a non-positive value is supplied.

range

The standard-deviation range for a type=sigma display. This defaults to 5.

key

If set to 1 (true), a colour table key is drawn alongside the displayed image. The default is 0, meaning no colour table key is drawn.

The order of play

Every time a primitive creates a meaningful file with a particular suffix, it sends a display request to the display system. For example, suppose the primitive that performs dark subtraction creates a frame called `f19990330_00042_dk.sdf` and then asks the display system to display it. The display system consults the *disp.dat* file for a dk entry. If no such entry is found, the display request is ignored and nothing happens. If one or more entries are found the display system proceeds to honour the request. If the *disp.dat* entry specifies a particular tool and/or window, the display system checks to see if they exist already and if not, stars them. Then it displays the data with the appropriate parameters.

Document info

Original author: frossie

C The ORAC-DR Calibration Selection

An ORAC-DR HOWTO

Description

ORAC-DR has a totally flexible system for controlling the automatic selection of calibration frames. This note describes how it works and how to override it

What happens

The type of calibrations used depend, obviously, on the instrument and the data reduction recipes used. Typically there are three kinds of calibration frames:

- Library frames provided by the observatory (bad pixel masks, rotation transformations, etc). These are maintained by the instrument scientist as appropriate. They are located in `ORAC_DATA_CAL`.
- Nightly frames that are generated during observing. The may be taken in specific calibration observations, e.g. by taking a "dark" (at UKIRT) or "skydip" (at JCMT) frames. They might also be generated from actual observations of targets (such as "sky flats") or calibration values (such as "flux conversion factors") calculated as part of a recipe. These are located in `ORAC_DATA_OUT`.
- "Rule" files that contain the rules for what constitutes an appropriate calibration frame. These are located in `ORAC_DATA_CAL`.

ORAC-DR treats the first two kinds rather differently.

Library frames reside `ORAC_DATA_CAL` and their selection is hardwired either in the instrument class or in a DR primitive. The users are unlikely to be concerned with them unless they want to override them with their own.

Nightly frames are handled in a more complicated way. A DR recipe that generates a calibration frame is responsible for filing it with the pipeline. The pipeline will hand it back to recipes that require calibration recipes according to a set of rules that are defined on a per-instrument basis by the ORAC-DR infrastructure as well as a per-frame basis by the calibration rules files.

Finding calibration frames

When a frame is reduced and files as a calibration, it is added to an index file located in `ORAC_DATA_OUT` named after the type of calibration, e.g., dark frames are filed in *index.dark*. When the pipeline is run up and needs a calibration frame but has not been asked to reduce one

in that session it will look in the index files for one that may have been reduced at a previous time.

If the pipeline is unable to find a suitable calibration it will complain vociferously and exit. This may seem extreme, but remember that ORAC-DR is designed for online use at an observatory. If an observer has not taken appropriate calibrations, we wish to point it out to them in the strongest terms because we do not want them to end up with un-reduceable data.

Overriding defaults

You can override the pipeline's selection of calibration frame by using the ORAC-DR `-calib` command line option. Use this override judiciously, as in general the pipeline does a fine job.

The ORAC-DR `-calib` command line option is used by giving comma separated key=value pairs (e.g. `'-calib dark=file1,bias=file2'`). The following keys can be used for general instruments. Specific instruments may have extra calibration overrides that can be used.

- baseshift - Use the given comma separated doublet (e.g. `"0,0"`) as the frame's base position.
- bias - Use the given frame as a bias.
- dark - Use the given frame as a dark.
- flat - Use the given frame as a flat.
- mask - Use the given frame as a mask. This option is usually used for bad pixel masks.
- polrefang - Add the given value to the measured polarisation angle to align the polarimeter's reference angle to north.
- readnoise - Use the given value for the detector readnoise.
- referenceoffset - Use the given comma separated doublet (e.g. `"0,0"`) as the frame's reference offset, which is difference between the frame centre and the reference pixel derived from the FITS headers.
- rotation - Use the given frame as a rotation matrix.
- sky - Use the given frame as a sky observation.
- standard - Use the given frame as a standard star observation.

When files are given the extension should be left off. As an example, if you have made a new bad pixel mask and wish to use it with ORAC-DR, the following command would be used:

```
oracdr -calib mask=new_bpm
```

D Shell Variables

Description

This document describes the complete list of environment variables used by the pipeline.

Complete variable list

ORAC-DR uses a number of environment variables for configuration.

User variables

Users may need to change the following variables before using the software.

`ORAC_CAL_ROOT`

Directory where the instrument-specific

`ORAC_DATA_CAL`

Location of the calibration files for the instrument. Set by the instrument startup script to `ORAC_CAL_ROOT/<instrument>`.

`ORAC_DATA_IN`

Actual location of raw data files. Should be set after the instrument startup script.

`ORAC_DATA_OUT`

Actual location of reduced data files. This is also the working directory of the pipeline. Should be set after the instrument startup script.

`ORAC_DATA_ROOT`

Location of the raw and reduced data directories if it conforms with the naming convention for the instrument. Should be set before the instrument startup script, which uses this variable to set `ORAC_DATA_IN` and `ORAC_DATA_OUT`.

`ORAC_HISTORY_DIR`

Location of the GAIA *.skycat_history* log. This is to avoid the monitor clashing with a running pipeline.

`ORAC_KAPVIEW_DEV`

If set, specifies a suffix to use for KAPVIEW device names. This can be used to have multiple runs of the pipeline use the same set of KAPVIEW windows. If not set, a suffix is generated based on the hostname and process ID to avoid conflicts between multiple instances of the pipeline.

`ORAC_KEEP`

If set, all intermediate files created by ORAC-DR will be kept. This does not include temporary files; to keep temporary files (*oractemp**) set `ORAC_KEEP` to "temp".

ORAC_LOCATION_DIR

Specifies a directory to which the location file should be written when processing data for the current day. (This file contains the location of ORAC_DATA_OUT and allows other systems to locate the pipeline's products.) If not specified, then "/jac_sw/oracdr-locations" is used.

ORAC_NOGROUPS

If set, group processing will be disabled.

ORAC_PRIMITIVE_DIR

Location of user-defined primitives. These supersede any identically names primitives in ORAC_DIR/primitives/<instrument>.

ORACDR_PROXY

If set, ORAC-DR will use a proxy for network lookups. This variable should be set to the full proxy name, including the protocol, name, and port. An example of this is "http://proxy.example.com:8181".

ORAC_RECIPE_DIR

Location of user-defined recipes. These supersede any identically names recipes in ORAC_DIR/recipes/<instrument>.

ORACDR_TMP

Location of scratch files for the ADAM messaging system. If this environment variable is not set, then this location will default to ORAC_DATA_OUT.

System variables

Starlink and EAO users should not redefine these variables under normal circumstances because they are correctly set by their user logins. They are included here for reference only.

ORAC_DIR

The location of the ORAC-DR software directory. This is normally set by a login script.

ORAC_INSTRUMENT

The instrument under whose environment ORAC-DR should run. Normally this is set by the appropriate instrument script in ORAC_DIR/etc/.

ORAC_LOOP

The default type of looping scheme that should be used for online reduction of ORAC_INSTRUMENT. This is used in the splash screen.

ORAC_PERL5LIB

The location of the ORAC-DR perl libraries. These are normally in ORAC_DIR/lib/perl5.

ORAC_PERSON

The e-mail address of the person who supports ORAC-DR for $ORAC_INSTRUMENT. This is used in the splash screen. (This was formerly the JAC contact name and assumed a JAC e-mail address $ORAC_PERSON@jach.hawaii.edu.)

ORAC_SUN

The Starlink User Note number that documents the data reduction for ORAC_INSTRUMENT. This is used in the splash screen.

Document info

Original author: frossie

E oracdisp

Control ORAC-DR display environment

SYNOPSIS

```
oracdisp [-h] [-v] [-in=file] [-out=file]
```

DESCRIPTION

Controls the display environment used by ORAC-DR. This routine can be used to edit the current environment and add display directives to the current environment.

ARGUMENTS

The following command line arguments are recognised by oracdisp:

-h

Prints help information describing the arguments.

-v

Prints version number information.

-in

Used to modify the name of the input display definition file. Default is *disp.dat*. Do not modify if used in conjunction with ORAC-DR.

-out

Used to modify the name of the output display definition file. Default is *disp.dat*. Do not modify if used in conjunction with ORAC-DR.

NOTES

By default, oracdisp manipulates a file called *disp.dat* in the ORAC_DATA_OUT directory. Do not modify the name of this file for use with ORAC-DR (since ORAC-DR is expecting a file called *disp.dat* in ORAC_DATA_OUT).

If ORAC_DATA_OUT is not set, and no overrides provided on the command-line, the program will try to find *disp.dat* in the current directory.

AUTHORS

Tim Jenness (t.jenness@jach.hawaii.edu), Frossie Economou (frossie@jach.hawaii.edu)

SEE ALSO

oracdr

COPYRIGHT

F oracdr_nuke

Kill all ORAC-DR related processes and shared memory

SYNOPSIS

```
oracdr_nuke
oracdr_nuke -nogaia -nogwm
```

DESCRIPTION

Attempt to kill all ORAC-DR related processes and shared memory that can be found and that are associated with the current user.

OPTIONS

-nogaia

Do not kill any GAIA processes.

-nogwm

Do not kill any GWM windows.

-help

Provide simple help information.

-man

Provides the manual page.

NOTES

- All shared memory owned by the current user is removed even if it is not directly associated with an ORAC-DR process.
- Will not attempt to remove shared memory owned by another user.
- Will attempt to kill processes owned by other users even though this will not succeed unless the user has special privilege.
- Does not attempt to clear out ADAM_USER directories. This is not normally a problem for ORAC-DR since each ORAC-DR process works in a different ADAM_USER directory.

AUTHORS

Frossie Economou (frossie@jach.hawaii.edu), Tim Jenness (t.jenness@jach.hawaii.edu), Alasdair Allan (aa@astro.ex.ac.uk), Brad Cavanagh (b.cavanagh@jach.hawaii.edu)

COPYRIGHT