Prospero Observer's Guide

A guide to observing using the *Prospero* package.

OSIRIS Edition
for the CTIO 4-m and 1.5-m Telescopes

Richard W. Pogge
The Ohio State University
Department of Astronomy

1999 February 17

Published by The Ohio State University, Department of Astronomy, 174 W. 18th Avenue, Columbus, Ohio 43210-1106

This manual has been composed in Times New Roman font using Microsoft Word97.

Prospero Observer's Guide

OSIRIS Edition for the CTIO 4-m & 1.5-m Telescopes

Richard W. Pogge
The Ohio State University
Department of Astronomy

1999 February 17

User Support

Limited user support for *Prospero* is available via email and the World Wide Web. This support only applies to problems with the *Prospero* package software proper. Problems with the instruments themselves must be referred to the appropriate instrument support personnel. Please report any software problems or bugs to the Prospero email support line:

`prospero@astronomy.ohio-state.edu`

Replies will be sent as soon as possible (this is not a support "hot seat").

World Wide Web

Information, additional documentation, a searchable online manual, sample scripts from this manual, and other resources for *Prospero* users are available at the Prospero Homepage on the World Wide Web:

URL: `http://www.astronomy.ohio-state.edu/~prospero`

Online OSIRIS support is available on a the Astronomy Dept. homepage at Ohio State:

URL: `http://www.astronomy.ohio-state.edu/~osiris`

A mirror site at CTIO is provided for observers at CTIO:

URL: http://www.ctio.noao.edu/instruments/ir_instruments/osiris/index.html

Information on the ICIMACS data-acquisition system is available at:

URL: `http://www.astronomy.ohio-state.edu/~isl`

Table of Contents

1 Introduction

Prospero is an interactive program used to make observations with CCD- or Array-based instruments from a Unix workstation. It serves as the front-end to the PC-based ICIMACS control system developed at Ohio State for use with our instruments and detector control systems. ICIMACS is described in separate documents on the ISL web site (http://www.astronomy.ohio-state.edu/~isl/).

Prospero presents the observer with an interactive command-language environment that provides all of the tools needed to acquire CCD or IR array data (both imaging and spectroscopy). Prospero provides commands for configuring the instrument (e.g., selecting filters, slits, etc.), defining exposure parameters (integration time, number of images in a sequence, object ID, etc.), acquiring data (starting single or multiple exposures), and simple telescope control (e.g., pointing offset and focus). When an exposure is completed, the data-taking system delivers a FITS format image file to a disk on the Unix workstation, and displays the image on a real-time video display. The observer can then load the image into their favorite image processing package (IRAF, VISTA, IDL, or whatever) to inspect the data or make measurements, copy it to tape, etc. Because it runs on a multitasking Unix workstation, the observer can maintain full control of the instrument and data acquisition system while at the same time having access to the wide range of image analysis tools available in their favorite image processing package, all on a single workstation console.

Prospero has two command modes: Interactive Commands and Procedure Scripts. In Interactive Command mode the observer types single commands into the Prospero window with the keyboard. Interactive Prospero commands come in two types:

1. **Low-Level Commands** to perform single, well-defined functions (e.g., the FILTER command).

2. **High-Level Commands** that combine a number of low-level commands operations into a single command to perform complex functions (e.g., the OBSERVE command).

Low-level commands make up the majority of the Prospero command set. They are based on the principle of "one command does one thing." The high-level commands are fewer in number, and take on the functions of one or more related low-level commands, thus reducing the amount of typing required for the most common observing tasks. The goal in designing Prospero has been to strike a balance between data taking programs that provide only low-level commands (e.g., older FORTH-based systems), and very sophisticated packages (like ICE or WildFire) in which high-level commands predominate. In this way, it can serve a wide range of observers from novices doing straightforward projects to experienced observers undertaking more complex projects or instrument builders working in the lab.

Procedure Scripts are mini-programs written in the Prospero command language to automate tedious or complex data-acquisition tasks. This is a very powerful mode of Prospero, and one you should take time to learn. Scripts allow an observer to "program" Prospero to automate their observing projects. For example, you can design a procedure script to acquire a multi-band imaging mosaic of an extended object that takes care of all instrument and exposure settings, moves the telescope to build the mosaic, sequences the exposures, and so forth automatically (i.e., following a "script"). While the Prospero script runs, you can be working in another window with IRAF (or VISTA or whatever) inspecting the incoming data as it is acquired, without interrupting the data acquisition process. Prospero scripts use the same commands as the interactive mode. In addition, *Prospero* provides a suite of specialized non-interactive commands for making Do-Loops, If/Then decision trees, prompting for user input, string manipulation, arithmetic, etc. Examples of scripts will be given in later sections of this guide, and are described in more detail in a separate document (*Prospero Command Procedure Scripts*).

These modes reveal Prospero's ancestry in the Lick Observatory VISTA package. While Prospero shares the command-language syntax and scripting mechanism of VISTA (and a few of its basic commands), it is important to emphasize that Prospero is not VISTA, nor is it an image-processing package like VISTA. Its role is to provide the observer with an intuitive, interactive environment for orchestrating the many different operations (control of instrument, detector, telescope, and data storage/archiving) that are essential to the process of observing with modern instruments. This *Observer's Guide* describes the basic commands that are used by most observers for routine observing. Instrument-specific features are covered in the specific instrument user's manuals.

This is a special edition for the system installed at the 4-m and 1.5-m telescopes of the Cerro Tololo Inter-American Observatory for use with OSIRIS. It describes special commands unique to OSIRIS, and uses examples drawn from use of this instrument. It is, however, not meant to be an OSIRIS manual. The complete OSIRIS manual is available online and in hardcopy form. Please consult that for detailed information and observing procedures.

2 Getting Started

This section reviews the startup procedures that need to be followed before you can start taking data with Prospero. **Important Note:** The following discussion assumes that the observing account has been properly initialized to run Prospero. See the detailed startup example in §8 for details.

2.1 Running Prospero

Prospero is executed either from the Unix shell with the command "prospero" (which assumes that an executable script so-named is in your path), or via a menu item on the X-windows console. Some observatory sites will automatically startup Prospero at login time. Upon startup, Prospero will occupy at least two windows on the workstation screen:

1. An Xterm used for command input

2. A status window which displays the current system status. On startup, this screen is mostly blank.

If all goes well, a welcome message is printed informing you that this is indeed Prospero you are running, and then it will load and run the default startup script that defines local aliases and internal setups. If successful, you should be rewarded with the generic Prospero command prompt:

```
1 PR>
```

The generic prompt means that you are in the lowest level of Prospero without the data-taking PCs connected or an instrument initialized.

The generic observing account setup at the SAAO 1-m telescope (observer) should start with Prospero running in an xterm immediately after logging in. If you need to restart Prospero during an observing session, it is available as a workstation menu item (available by clicking on the left mouse button while the cursor is on the console background).

2.2 QUITting Prospero

The QUIT command ends a Prospero session. To prevent accidental exits, Prospero will ask you to confirm (by typing Y or N) that you really want to quit. For example:

```
PR> quit
PR> Do you really want to QUIT Prospero <y|n> [n] ? y
PR> STOP: Prospero done
```

The text "y" typed after the "?" was typed by the user in response to the prompt.

Note that because the instrument and detector systems are controlled by independent computers, accidentally quitting Prospero does not (necessarily) have any bad effects. If Prospero dies while an integration is in progress, you will probably get some extraneous error chatter, and in any event, you will have to wait until it is finished to restart Prospero and reconnect. If everything is working correctly there are safeguards to help ensure that your data won't get lost should Prospero crash or you quit prematurely.

If running Prospero from the observer account (either from the session started automatically on login or invoked from the mouse menu), the Prospero xterm and status windows will both disappear when the program terminates.

2.3 Connecting to the Data-Taking PCs

Before you can begin taking data, you need to connect Prospero to the data-taking PCs, and perform a few basic initialization chores related to the instrument control and detector systems.

To connect Prospero to data-taking PCs, type

```
PR> STARTUP
```

If things go well, you should see a screen full of startup chatter between Prospero and the various processes, and the Prospero Status window will begin to be filled with detector, instrument, and system status information. A detailed startup description is given in §8, where a typical startup sequence is dissected and possible error conditions explained.

2.4 Starting a Night's Observing

After connecting to PCs, the last step is to execute the RUNINIT command and answer the questions. For example:

```
2 PR> runinit
Observing Session Initialization
Initialize Session <y|n> [y] ?
Enable Writing Files to Disk <y|n> [y] ?
Image directory path [/ua12/iraf/osiris] ? /ua14/iraf/osiris
Image Filename (rootname.nnn) [991010.354] ? 9901011.001
Observers ? Larry, Curly, & Moe
Comment ? Night 2 of 3
Save setup? <y|n> [y] ?
```

As before, any input after the "?" was typed by the user. If there is nothing after the "?", then the user has hit the Return key and accepted the default value appearing within the []'s.

RUNINIT establishes the basic runtime defaults that will be used for the night's observing. These defaults will be restored should Prospero or the PCs need to be restarted during the night. It is important to issue RUNINIT at the start of each night to make sure the basic defaults are set correctly. If you neglect to go through this step, you could end up with "stale" defaults from the previous night (or even from someone else's run.

3 Basic Observing Commands

This section will assume that Prospero has already been successfully connected to the data-taking PCs, and that you are ready to begin taking data.

3.1 Observing with OBSERVE

The OBSERVE command is patterned after the command of the same name and function in the ICE system used at NOAO (Massey et al. 1992). OBSERVE steps observers through the basic questions that they should always ask themselves before actually starting a new observation. It takes on the functions of a number of individual low-level commands in a systematic way. For example:

```
PR> observe
Image type (object|zero|bias|dark|flat|comp|std) [flat] ? object
Object Name [J Dome Flat] ? MB-98-123 H
Integration time (seconds) <0.:7200.00> [1.] ? 60
Number of images to acquire <1:99> [10] ? 3
Number of co-adds/image <1:99> [5] ? 3
Doing setup...
Exposure Sequence Started.  Hit X to abort
STATUS: Finished  1 of  3 exposures.
STATUS: Finished  2 of  3 exposures.
STATUS: Finished  3 of  3 exposures.
DONE: Exposure Sequence Completed.
```

In this example, the observer's responses are typed after the "?" and if there is nothing after the this means the default given in the was accepted by hitting Return. After the last prompt, we see the setup and exposure execution chatter. The defaults are taken either from the last settings used (either set with individual commands or the last OBSERVE command).

Let's dissect this example into its components:

```
Image type (object|zero|bias|dark|flat|comp|std) [FLAT] ? object
```

asks for the type of image to be acquired, with the list of possibilities in the ()'s. In this case, the previous image was a flat field, and the observer has selected the next image to be an image of an object. Note that the response is case-insensitive. Image types are explained in §4.3.1 below.

```
OBJECT Name [J Dome Flat] ? MB-98-123 H
```

asks for the name of the object. Note that ()'s and single-quotes (') are reserved characters in the data-taking software running on the PCs. If you type them, Prospero will change ()'s to []'s, and single-quotes to blank spaces.

```
Integration time (seconds) <0.:7200.00> [1.] ? 60
```

asks for the integration time in seconds. Here, the last image used a 1-second integration time, and the observer has changed it to 60-seconds. Note that the decimal point (.) may be omitted.

```
Number of images to acquire <1:99> [10] ? 3
```

asks for the number of images to be acquired during this OBSERVE command. In this example, the previous exposures were taken in groups of 10, and the observer has changed it to 3. This means that OBSERVE will take 3 exposures of 60 seconds each in succession, storing them as 3 separate images. This is a way to take multiple images in succession without having to enter the same information over again (increasing your observing efficiency.

```
Number of co-adds/image <1:99> [5] ? 3
```

Asks you for the number of separate integrations that are to be co-added and then averaged into a single image. The previous image used 5 co-adds/image, and here the observer has requested 3 co-adds/image (i.e., each "image" acquired will be the average of 3 of the given integration time). Co-addition is a common strategy for building up signal-to-noise with IR array detectors given the short integration times required to avoid saturation in a high-background environment.

At this point, OBSERVE has all of the information it needs to start the exposure sequence. It first does any low-level setups required (including filter changes), that it indicates by printing:

```
Doing setup...
```

and then it starts the requested exposure sequence:

```
Exposure Sequence Started.  Hit X to abort
STATUS: Finished  1 of  3 exposures.
STATUS: Finished  2 of  3 exposures.
```

and so forth until the sequence is completed (or interrupted by the observer).

Note that settings made with OBSERVE are "sticky" in that they will become the defaults for the next OBSERVE command, or until you change them with the individual low-level commands represented by each question. If instrument configuration changes are required, the OBSERVE command makes the necessary changes before the exposure sequence is started.

An option of OBSERVE is to have it review the instrument configuration (filters, cameras, slits, etc.) after getting the exposure parameters. This feature can be enabled with the OBSPARS command that is used to set the default behavior of OBSERVE.

3.2 Repeating Observations with REPEAT

You can repeat the last exposure sequence with REPEAT. An optional argument will let you repeat the last sequence as many times as you wish, for example:

```
repeat 5
```

repeats the last exposure sequence 5 times. This means that if the last sequence took 4 images in succession, this command will take 5 sequences of 4 images each for a total of 20 images! It really does mean "do exactly what I just did." REPEAT makes no changes to the instrument or integration settings.

REPEAT can save you having to go through the interrogation of OBSERVE unnecessarily if nothing is to change except that you want to take more of the same. However, please note that REPEAT is not an efficient emulation of the low-level MGO command. MGO exploits certain hardware timing features, especially with IR arrays, that minimize dead time between images, whereas REPEAT simply re-issues the exposure commands in a loop, with all the attendant overhead. It is a better idea to plan ahead when you issue OBSERVE (if possible), or if you want to take more data, to use the MGO command instead. REPEAT is more mnemonic than MGO for some users.

You take can advantage of MGO for sequences of images (with the same instrument setting) through OBSERVE by judicious use of the "Number of images/sequence" parameter (see §3.1). Observers who prefer to use the low-level exposure commands (GO, MGO, etc.) will discover that REPEAT works with them as well, in the same way.

3.3 "What's in a name?"

Raw images are written in FITS format with filenames of the form:

```
rootname.nnn
```

where "nnn" is a 3-digit number giving the order in sequence that the image was taken. Image numbers run from 000--999. The rootnames of files are case insensitive, and capital letters are rendered in lowercase on Unix disks.

If the files are being written to a Unix disk, the rootname may consist of letter and numbers, but must not contain any dots (".") or special characters except the underscore (_). The extension must *always* be a 3-digit number. If (for some reason) you must write data to one of the DOS disks on the data-taking PCs, the filename must follow the strict DOS 8.3 convention. This is not required by Unix systems, but you should keep filenames simple.

A good choice for an image filename is the UTC date in YYMMDD format (e.g., 990622). This gives you 1000 files (990622.000 through .999) with names that encode the date of observation and that are likely to be unique.

If the requested filename already exists, the data-taking system will automatically switch to a fail-safe "unique name" convention to prevent accidental data destruction. If you start getting files with names like "991203os.03g" on the disk (or in the "Lastfile:" entry on the status display) it means you've triggered the fail-safe condition. See §4.4.2 for a discussion of the unique-name conventions used by the system and how the fail-safe condition is triggered.

3.4 Interrupting an Integration

While an integration is counting down, control is handed over to the PC with the detector controller. This PC is responsible for the real-time task of timing the integration and reading out the detector. As a consequence, you cannot issue other Prospero commands during an integration, but you may of course work in another window with IRAF or whatever looking at the data you've already taken or that is streaming down onto the hard disk from a multi-image sequence started with OBSERVE.

To exercise some control over an integration, the observer is given a limited number of single-key commands that interrupt the integration should things go awry.

Because staring IR arrays are unshuttered and rely on differences between image (post- and pre-reads) to create an image, it is impossible to pause an integration or restart it. Thus the only option available with OSIRIS is the "X", which will abort the integration and discard it if things go wrong. If the "ABORT" signal arrives within about 5 seconds of countdown termination, it may not be caught, so as a general rule of thumb do not attempt to abort an integration that is within about 20 seconds or so of the end of the exposure. If you are in the middle of a sequence of images (e.g., an MGO), you may issue an "X" to abort at any time. It should catch it on the next image, if not the current exposure if there is enough time to process it before the countdown finished.

Patience is essential. Once you issue the ABORT directive, the system needs to take care of the abort and clean up after itself. All of this takes time. The system does not respond to an ABORT request immediately. **BE PATIENT.** If you hit X many times, or try to Ctrl-C out (see §3.5 below), you can cause serious problems.

3.5 Ctrl-C is not always your friend...

The Ctrl-C key can be used to interrupt a Prospero command, an executing procedure script, or to cancel pending communications requests between Prospero and the PC system.

If you attempt to abort an integration by hitting Ctrl-C, Prospero will issue the abort and initiate the clean up, eventually returning you to the interactive command prompt. This is not the recommended way to abort (the X key is), but if all else seems to fail (for example, if X fails to get an abort acknowledgment), then try Ctrl-C, but please remember this one simple rule:

ONLY HIT CTRL-C ONCE AND WAIT.

Prospero uses a Ctrl-C handler to catch when the observer hits the Ctrl-C key, and signals to all active routines that an interrupt has been requested by the observer. The interrupt handler then sets in motion a series of recovery/clean-up procedures to ensure that Prospero and data-taking PCs are not left in an ugly state. If you hit Ctrl-C repeatedly, especially many times in rapid succession, you could interrupt the clean-up routines and cause bad things to happen.

To fully recover from a Ctrl-C interrupt, issue a FLPR command. FLPR is analogous to, but functionally quite different than the IRAF/ICE flpr command.

4 General Prospero Commands

High level commands like OBSERVE are only part of the story. This section briefly describes the suite of low-level Prospero commands that will give you greater control over the instrument and various data-taking functions than the basic observing commands described previously. Complete descriptions of these commands are available via the online HELP utility.

4.1 Getting HELP

Prospero has an extensive online help manual with a command search utility. To read the help file for a specific command, you type:

```
help go
```

This will print the online help file for the GO command on the terminal screen, using the default pager (usually the Unix more command, see §5.2) to allow you to scroll backwards and forwards through the document.

If you cannot remember the name of a command, but have some idea as to its function, you can search for the appropriate command using APROPOS. For example:

```
apropos expos
```

will print a list all commands for which the string "expos" appears in either the command name or its description. Note that this search string is case insensitive. You can then select possibilities from this list to explore further with HELP.

If you are not sure where to start, typing HELP without arguments will put you into interactive help mode. This will display the table of contents of the online manual, permitting you to select the category of interest (e.g., Instrument Control), and then to select from among the entries under that category. A Web-based hypertext version of the online manual is also available with an APROPOS-like search engine. Some sites have their own copies of this, see your facilities manual or type HELP ONLINE in Prospero for details.

4.2 The Prospero Status Display

On a workstation console running X-windows (or an equivalent like Sun's OpenWindows), Prospero keeps a running display of all system status in a separate Status Window. The status window shows the state of connections to the data-taking PCs, image parameters (title, image type, exposure time, etc.), filename info, instrument status (e.g., filter, prefilter, slit, etc. as appropriate), and any other information appropriate to the instrument you are using. The instrument manuals have examples of typical Prospero status displays.

Status is updated each time changes are made to system parameters, and the update time (in UT) is displayed in the lower left-hand corner of the window so you can tell if the status display is stale. You can force the status window to update with the following commands:

`STATUS`: queries all system computers for their current status
`ISTATUS`: only queries the instrument for its status.

In addition, the `CLEAR STATUS` command can be used to refresh the status window display (sometimes it gets "damaged" interacting with other windows on the console), but it does not update the information.

A simple color scheme is used to make it easy to assess the overall system status at-a-glance. In general,

YELLOW = "this parameter is within the valid range"
GREEN = "Flag is On" or "Condition Normal"
RED = "Flag is Off", "Error Condition" or "Warning"
GREY = "Device is out of the beam"

We tried making all "parameter valid" values green, but it proved to be very hard on the eyes. As such, we chose green for the normal state of system flags (which are either on or off, valid or invalid), and yellow for user- and system-defined parameters which take a range of values.

For system parameters (like filenames, device positions, etc.), the fixed parameter labels are shown in WHITE, with their current values displayed in YELLOW if normal, or in RED as asterisks (`*`) or asterisks with a question-mark (`*?*`) if the parameter is unknown or in error.

On OSIRIS, some instrument mechanisms (the prefilter wheel, pupil mask, and grating) can be inserted into or retracted from the beam. In these cases, if a device name and its parameters appear gray, this means that mechanism is no longer in the beam. This provides a way of telling at a glance what the instrument configuration is, while still being able to see the settings of otherwise irrelevant mechanisms (irrelevant here meaning that it doesn't matter what the prefilter is if the prefilter wheel proper is not in the beam).

There are various system "function flags" that are displayed as a flag name with a + or - sign to indicate whether this function is enabled or disabled. For example:

`+DISK` means disk writing is enabled
`–BELL` means that exposure bell has been disabled

Flags are colored GREEN if things are normal, and RED if abnormal, where "normal" in this context means that this function is required for normal data acquisition or instrument control. For example, `–DISK` is red to remind you that any data acquired will not be stored to disk, and `+MOVIE` is red to indicate that the detector is in continuous read mode and neither image storage nor instrument control is available. Functions that have no bearing on normal operations are always green (e.g., `+BELL` or `–BELL` are always green). Flags and their meanings depend on the instrument. See the relevant instrument manual for details.

4.3 Data Acquisition Commands

Prospero provides a number of low-level commands that form the basis of the high-level OBSERVE command (§3.1). These fall into three categories: setting the image type and title, setting the integration time, and starting exposures.

4.3.1 Image Type and Title

The image type (object, flat, dark, zero, etc.) and the object title are set at the same time in Prospero via the image type commands. The generic command syntax for setting the image type and title is:

```
imagetype 'a title for the object'
```

Where imagetype is one of

OBJECT	astronomical objects
FLAT	flat field images
SKY	sky images
COMP	comparison lamp images
STD	standard star images
DARK	dark images (with a dark plug or mask put in the beam)
ZERO	zero-second bias image
BIAS	common alias for ZERO images

and the object title is given as a string enclosed in single quotes. Thus

```
flat 'Dome Flat @ K'
```

Will set the image type for subsequent images to flat, with the title “Dome Flat @ K”. Note that the single quotes are delimiters used by the command, and do not appear in the title itself. Image titles may consist of letters and numbers, and some special characters (e.g., @, _, etc.), but the data-taking PCs treat parentheses, (), and the quotes (both ' and “) as reserved characters. If you type them, ()’s will be converted to []’s, and quotes will be discarded.

The imagetype code is stored in the IMAGETYP header card, and the title in the OBJECT header card. If an image type command is given without arguments, the IMAGETYP card is changed, but the title remains the same. Note that the BIAS and ZERO image types force the exposure time to zero seconds. The previous exposure time is saved so it can be restored when other image type commands (OBJECT, DARK, etc.) are given.

Note that DARK frames on OSIRIS require that you separately put in any dark blocking plugs (in the filter & prefilter wheels) and/or dark masks yourself. DARK only sets the keyword in the header.

4.3.2 Changing the Integration Time

EXPTIME sets the integration time in units of seconds or minutes. For example:

```
exptime 20
```

sets the integration time to 20 seconds, whereas

```
exptime 20 minutes
```

sets the time to 20 minutes. Units of seconds are assumed by default if no units are given on the command line.
Integration times are rounded off to the nearest tenth of a second, thus entering 0.52 or 0.48 with exptime will set the integration time to 0.5. The shortest integration time allowed for shuttered CCDs is 0.1 seconds, but the true exposure time for less than about 0.5 seconds depends critically on the shutter and timing electronics. The true minimum integration time for IR Arrays similarly depends on the details of the readout electronics. Consult the relevant instrument manuals for your specific detector system.

4.3.3 Taking Exposures

Prospero can take exposures singly or in sequences of images taken one after another, or it can take sequences of exposures that are co-added on-the-fly into a single image that is finally written to disk. There are four exposure commands:

`GO`	`take a single image`
`MGO m`	`take a sequence of m single images`
`AVEGO n`	`take n images and coadd them into a single image`
`MAVEGO m n`	`take a sequence of m images of n coadds each`

Each of these commands uses the current exposure time and image type. The abort modes for each of these exposure types behave slightly differently, as follows:

GO aborts the current image and resets the system. No image will be written to disk.

MGO aborts the current image in the sequence, discards it, and then stops the sequence and resets the system. Only those images already taken in the sequence will be stored.

AVEGO aborts the current image in the sequence, discards the running sum, and resets the system. No images are written.

MAVEGO aborts the current image in the sequence, discarding the current running sum, then stops the sequence and resetting the system. Only previously completed co-added images are written to disk.

See §3.4 for a more general discussion of how to abort integrations.

4.3.4 Relation to OBSERVE

The OBSERVE command described previously sets up an exposure sequence by asking questions like this:

```
PR> observe
Image type (object|zero|bias|dark|flat|comp|std) [flat] ? object
Object title [Dome Flat @ K] ? NGC 1068 @ J
Integration time (seconds) <0.:7200.00> [1.] ? 30
Number of images/sequence  <1:99> [10] ? 5
Number of co-adds/image  <1:99> [5] ? 3
```

The same operation would be accomplished by issuing the following three Prospero commands:

```
object 'NGC 1068 @ J'
exptime 30
mavego 5 3
```

Use of these low-level commands is more appropriate in the context of Prospero command scripts than high-level, highly interactive commands like OBSERVE.

4.4 Image File Management

The data-taking system write a FITS format image file with your data onto a hard disk mounted on the Unix workstation. At this point, the data-taking system's job is finished, and you can store the image to tape or process it further with the image processing package of your choice (IRAF, VISTA, or whatever) as you wish. Prospero performs no post-processing or further manipulation of the data.

4.4.1 The Image Directory

On startup, data-taking PCs mount one or more Unix file systems for storing data in FITS format files. The image directory can be changed using the PATH command (or its alias, IMDIR). For example:

```
path /ua12/iraf/osiris
```

tells the data-taking PCs to write images to the Unix disk file system "/ua12/iraf/osiris". The PC programs will test the requested directory path for validity and will return an error message if that file system is not mounted. The OSIRIS manual has an updated list of the available raw data disks that you may use, since these tend to change with time.

4.4.2 Image Filenames

Raw data are written to disk in FITS format with filenames of the form:

```
rootname.nnn
```

where "nnn" is a 3-digit number giving the order in sequence that the image was taken. Image numbers run from 000--999. **The file extension must be a number.** The number of the next image in the sequence can be changed without changing the rootname with the NEWEXT command. See §3.3 for a discussion of the Prospero filename convention.

The name of the last file written to disk and the next file to be written to disk after the next exposure is displayed in the Status window. You can also print this information the screen using the LASTFILE and NEXTFILE commands. Note that these commands only query the system for information and do not make any changes to the filenames.

To prevent the possibility of accidentally overwriting an existing data file, the system employs assigns each image created a coded "unique name" (assigned to each image upon creation). In the even to try to overwrite an existing file, the system switches to the unique name.

The unique name is of the form:

```
990623os.03a
```

It is composed of the UTC date (YYMMDD) followed by unique 2-letter instrument ID code (e.g., "os"=OSIRIS), and the extension is a 3-character, base-36 coded string composed of letters and numbers ([a-z,0-9]) in ASCII order (the ".03a" in the example). This convention provides up to (46656) unique filenames for a given UT-date. A unique name is assigned to each image and stored in the UNIQNAME header card whether it is actually used or not.

4.4.3 Enabling and Disab1ing Image Writing

Disk storage of images is automatically enabled on system startup. If enabled, you will see a green `+DISK` flag on the status display. If disk storage is disabled, you will instead see a red `-DISK` flag. Disk storage can be enabled or disabled at any time using the commands:

```
+DISK
```

and

```
-DISK
```

respectively. For users familiar with the old OSU FORTH data-taking system used with the original OSIRIS, the aliases `+STORE` and `-STORE` are provided to perform the same functions as in that system.

Note that disabling disk writing can be dangerous – you could forget to turn it back on and lose valuable data (the system will not complain if disk writing is turned off!). In general it should only be done if absolutely necessary (e.g., if doing engineering checkout and you don't want a semi-infinite number of garbage images sent to the Save-The-Bits archiver). Be careful!

4.4.4 Disk Space Issues

OSIRIS images are 1024×1024 pixels in size, and if co-added are written as 32-bit floating point numbers per pixel, resulting a 4Mb data FITS file. On SunOS systems, the largest typical disk file system is only ~1.6Gb in size, which will fill up after ~400 images are written by OSIRIS. If the disk is filled during data-acquisition, a fatal error message is sent to abort further data acquisition until space can be found to write the data. This requires the observer to take one of the following actions:

1. Make room on the current data disk by deleting junk files or data already archived to disk (make sure you confirm the latter!). If you are processing data on the fly with IRAF, junk files can add up fast.

2. Change the image data directory to a new disk with more space available using the PATH command.

Recovery from overwriting a disk is difficult, and requires that you reboot the data-taking PCs, restart the caliban daemon, and restart the Prospero session. This is painful and costs observing time. The alternative is to keep an eye on your disk space use and switch to one of the other data disk when the current disk is more than 95% full. Disk space management is the responsibility of the observer.

4.5 Instrument Contr ol

Prospero provides a wide suite of instrument commands. Since this is a general guide to observing with Prospero, we do cannot review all supported instruments here, please consult the OSIRIS User's Manual or the Quick Reference card for details. Copies of each are available online.

In general, instrument control commands are of the form:

```
PR> filter 5
```

where the command name is the name of the particular device (e.g., `filter` for a filter wheel), and the argument is the number of the desired device position. For discretely positioned devices, like filter select mechanisms, this number is a particular device position. For continuously positioned devices, like focus mechanisms, the number usually refers to an encoder value for the desired mechanism position, and lies within some specified range.

For instrument mechanisms like filter and slit wheels that are populated with optical elements or devices (both installed by the instrument builder or by the user, like in the case of user filters), Prospero stores internal tables of the current device populations. You can view these tables using the PRINT command, for example:

```
print filter
```

will show the currently (known) populations of the filter wheel(s) in the current instrument..

4.6 Detector Control

Prospero provides a limited number of user commands for direct control of the detector readout (in addition to a restricted suite of engineering level commands not discussed here).

The basic detector commands common to IR arrays and CCDs are as follows:

MOVIE (+MOVIE)	Commands the detector computer to put the array into continuous readout mode, taking a continuous set of exposures at the current integration time. Images are displayed on the IC's real-time display as they are acquired, but not stored anywhere. While the detector is in MOVIE mode, you may only change the integration time (EXP command).
STOPMOVIE (–MOVIE)	Gets the detector out of MOVIE mode, and returns to normal data-acquisition mode.
ICSCALE	Used to change the intensity display scaling parameters used by the IC's real-time display. The command syntax is `ICSCALE min= max= sat=` where: `min` and `max` set the minimum and maximum data values displayed (min as black, max as white, gray between), and `sat` sets the saturation threshold, turning all pixels greater than this red to show saturated pixels.

4.7 Telescope Contro1 Commands

The complete remote command set of the CTIO Telescope Control System (TCS) is accessible from within Prospero. This section describes the main TCS commands provided for observing with OSIRIS. For other commands, see the TCS manual in the control room, or look at the online help entry "CTIOTCS" in Prospero.

4.7.1 TELSTATUS: Return the current TCS Status

TELSTATUS returns the current Date, Time, and pointing information from the TCS. The output is formatted for ease of reading. Note that some entries are blank as the CTIO TCS does not report all possible TCS information at the present time.

4.7.2 OFFSET: Offsetting the Telescope

Prospero provides the following commands for offsetting the telescope:

`OFFSET ra= dec=`	Offset the telescope in both RA and Dec (entries in arcseconds)
`NORTH xx`	Offset the telescope xx arcseconds North
`SOUTH xx`	Offset the telescope xx arcseconds South
`EAST xx`	Offset the telescope xx arcseconds East
`WEST xx`	Offset the telescope xx arcseconds West

Note that all RA offsets are in units of arcseconds projected onto the sky (in “plate” or “normal” coordinates), and do not need to be corrected for cos(δ). Hence “EAST 10” will offset the telescope 10 arcseconds east of the current position, not 10×cos(δ) arcseconds.

4.7.3 TELFOCUS: Change the Telescope Focus

TELFOCUS will instruct the TCS to change the secondary mirror focus to the desired value. This command permits relatively simple focussing of the telescope onto the instrument focal plane, and can be used to automate focussing using a simple Prospero command script.

4.7.4 LAMP and MIRROR: View Comparison Lamps

These commands are only available at the 4-meter telescope (the 1.5-m does not have spectral comparison lamps).

LAMP is used to turn on the comparison lamps in the 4-m guide/acquire module. A summary of the lamp commands is as follows:

`LAMP 1 on`	Turn on the HeAr Lamp
`LAMP 2 on`	Turn on the Neon Lamp
`LAMP 1 off`	Turn off lamp 1 (HeAr lamp)
`LAMP OFF`	Turn off all lamps that are on.

In order for OSIRIS to view the comparison lamps, it is necessary to insert the pickoff mirror into the beam. This is done with the MIRROR command:

MIRROR IN	Insert the pickoff mirror into the beam (view lamps)
MIRROR OUT	Retract the pickoff mirror (view the sky)

A simple rule of thumb for how to use the MIRROR command is:

“When in doubt, MIRROR OUT.”

Note that “IN” corresponds to mirror position 4, and “OUT” corresponds to position 1. OSIRIS does not use the other two mirror positions.

4.7.5 GUIDER Commands

Three simple commands are provided for controlling the autoguider from Prospero at the CTIO 4-m telescope. The autoguider commands are as follows:

`+GUIDER`	Enable (resume) guiding
`-GUIDER`	Disable (pause) guiding
`GUIDER RA=hh:mm:ss DEC=dd:mm:ss`	Move the guide probe to (RA,DEC)

No autoguider is available for use with OSIRIS at the CTIO 1.5-m f/13.5 focus.

4.7.6 Beam Switching

The CTIO 4-m TCS also permits simple 2-point beam switching accessible via two Prospero commands:

`BSWSET RA=xxx DEC=xxx`	Set the offsets (in arcseconds) for the "B" beam position. Beam "A" is the current position.
`BSWITCH [A|B]`	Switch to position A or B

Beam switching is not available at the 1.5-m telescope.

4.8 Operating System Commands

To issue an operating system command from within Prospero, use the $ command. There are two modes:

1. Typing $ by itself will do a "shell escape", landing you in the Unix operating system with a generic c-shell. You can enter Unix commands as you wish. To return to the Prospero prompt, type the Unix exit command.

2. Typing $ followed by any valid Unix command will execute that command and return to the Prospero prompt when done. Thus, typing "$ ls" will issue the Unix ls command in the current working directory.

To make life easier, many people define common Unix shell commands (like ls, df, etc.) in their startup scripts. See the example in the *Prospero Command Procedure Scripts* guide, or just type `ALIAS` to see if your local default installation of Prospero has pre-defined these in the default startup script for your site.

5 Customizing Prospero

Prospero offers a number of commands that allow you to customize its run-time environment. This includes defining command aliases to save typing, selecting your favorite file pagers and editors if the defaults are not to your taste, and to enable special features like warning bells.

5.1 Command Aliases

The ALIAS and UNALIAS commands are used to define and remove aliases for commands (including command arguments). Judicious use of aliases can save you typing if you have several commands that will be used repeatedly. Bob Blum has set up a large number of pre-defined aliases especially for OSIRIS. These are described in the online instrument manual.

There are two ways to define an alias. The first is to use the full command syntax. For example:

```
ALIAS F1 'FILTER 1'
```

This defines a new command, F1, which executes "FILTER 1" when it is typed. As usual, commands and aliases are case insensitive, so F1 and f1 would both be recognized. However, beware that arguments included as part of a command alias will be always executed in the in the case in which you typed them. Note that if the alias is to be composed of more than one word, the entire command must be enclosed in single quotes.

The second way is to only type the new name of the new alias, and have the program prompt you for the definition, thus:

```
PR> alias wide
Enter an alias for WIDE >>
```

You can then type the alias for WIDE at the prompt (the >>'s).

Typing ALIAS with no arguments will print a list of the currently defined aliases.

You can add keywords to the command at the time of execution by typing them after the synonym. For example, suppose you defined the alias "T" to be:

```
ALIAS T 'TV 1 Z=0 L=1000 CLIP'
```

If you were to type the command:

```
T BOX=1
```

it will execute:

```
TV 1 Z=0 L=1000 CLIP BOX=1
```

Appending the "`BOX=1`" argument to your alias.

Aliases need not be commands that would actually run by themselves. For example:

```
ALIAS F 'FILTER'
```

is a valid alias. Typing F by itself as a command would not work, since FILTER is not a complete command (it needs an filter wheel position number). However, if you provide it with an argument, like:

```
F 3
```

the filter wheel will to position 3.

The only restrictions on aliases are as follows:

1. You cannot use aliases inside other aliases.

2. You cannot define an alias that executes multiple commands chained together by semicolons ('). **One command per alias.** If you want an alias to execute more than one command at a time, you should write a command script instead (see §6).

UNALIAS command removes aliases. For example:

```
unalias f1
```

deletes the F1 alias we defined as the first example in this section.

You can have your favorite aliases defined automatically each time you start Prospero by putting ALIAS commands into the startup procedure script, or by putting them in your own personal aliases script and executing it with the CALL command. See the *Prospero Command Procedure Scripts* guide or the relevant online help files for details.

5.2 Pagers and Editor s

The Prospero HELP and NEWS commands use generic Unix screen pagers like more (or less) to display large text files. This allows the reader to scroll backwards and forwards, search for words, etc., making the files easier to read. To change the default pager, or get information about it, use the SETPAGER command. Typing the command:

```
SETPAGER
```

without arguments will print the name (usually the full path and executable filename) of the current default pager. Two common and popular pagers are selectable via keywords:

```
SETPAGER more
```

will select the standard Unix more pager, and

```
SETPAGER less
```

will select the less pager, assuming the path is /opt/local/bin/less as per the usual installation instructions. Some system managers like to be creative, so you may have to use the PAGER= keyword to give the explicit path to the less executable. See the online help file for SETPAGER for details. Note that any changes you make to the default pager will be forgotten at the end of a session. You might consider if you want to use a custom pager to put a SETPAGER command in your personal startup script.

All of the editing commands within Prospero (PEDIT, HEDIT, and EDIT) use generic Unix screen editors like vi or emacs (no sense reinventing those particular wheels). By default, Prospero assumes the common vi editor. You can change the default editor with the SETEDITOR command. Typing the command:

```
SETEDITOR
```

without arguments will print the name (usually the full path + executable filename) of the current default editor. Two common and popular editors can be selected via keywords:

```
SETEDITOR vi
```

will select the vi editor (/bin/vi), and

```
SETEDITOR emacs
```

will select the GNU Emacs editor, assuming the usual path. The EDITOR= keyword can be used to select other editors, for example:

```
SETEDITOR editor=/usr/openwin/bin/textedit
```

selects the OpenWindows textedit tool as the default editor. See the online help file for SETEDITOR for details. As with the pager, any changes made will be forgotten at the end of the session, so consider inserting a SETEDITOR command in your startup script if you don't like the defaults.

5.3 Bells and Whistles

Prospero can be made to make noise when it wants your attention. You can also turn these noises off if you don't want more quiet operation.

The Exposure Bell will ring when an integration is completed. This is useful for getting your attention when things are finished, but if you like you can toggle the exposure bell on or off using the `EXBELL` command. By default, the exposure bell is always enabled.

There is a prompt bell that will ring the bell every time the command prompt appears. Some people like this. Other people are driven insane by it. You can toggle the command bell on or off, using the `BELL Y` and `BELL N` commands, respectively. By default, the command bell is always OFF.

If you just want to make noise, type "`BELL`"

There are also commands available for ringing bells within procedure scripts to wake up the observer to do something during (or after) a long observing procedure, see ALERT and WAIT in the Prospero Command Procedure Scripts guide for details.

6 Prospero Procedure Scripts

6.1 Overview

Prospero normally works by executing individual commands typed at the keyboard. It is also possible to execute a list of commands that are stored in an external text file. These "Command Procedure Scripts" are one of the most powerful features of Prospero. It allows you to customize Prospero to perform repetitive or complex tasks by writing mini-programs in the Prospero language, rather than having to write (and debug and install) a new Fortran subroutine module for each new task.

At the very least, a procedure script can be nothing more than a list of the Prospero commands in the order that would have typed by hand. The difference is that these commands are stored in a file and executed as a procedure instead of being typed in line-by-line (i.e., it is literally a "script" of commands to be followed by the program). Scripts can make your life much simpler.

In addition to simple "command list" scripts, Prospero provides a set of specialized functions for advanced procedure control. These include DO-loops, IF/ELSE decision trees, branching, input prompting, message printing, string manipulation, external file I/O, and various types of PAUSE and SLEEP directives. These commands greatly expand the possibilities of procedure scripts to the creation of "programs" written in the Prospero command language. This makes Prospero user-extensible and greatly enhances its utility. A detailed discussion, including a list of the various scripting commands, may be found in the separate *Prospero Command Procedure Scripts* document.

Bob Blum and others have setup a fairly extensive suite of observing scripts to assist OSIRIS observers. Many replicate the functions of tcl scripts developed for CIRIM and IRS, and users of those instrument will find them familiar and useful. Writing your own scripts, however, is a great way to optimize your use of OSIRIS.

6.2 Example Scripts

The following are examples of simple Prospero scripts that are appropriate for first-time users. No doubt you will find that your colleagues have written scripts that are generally useful and available on your system. The examples below are meant as introductions, and are neither exhaustive nor the only ways to do things, but they should give you a flavor of how to use scripts to make observing easier.

6.2.1 An Image Sequence (Part I)

This is a simple "line-by-line" script that does the following:

1. Take 5 images of 30-seconds each through a J-band filter in filter wheel slot 1.
2. Take 3 images of 60-seconds each through the H-band filter in slot 2.
3. Take 5 images of 40-seconds each through the K-band filter in slot 3.

The object title is changed each time to reflect the new filter, and the MGO command is used to acquire the multiple integrations. This script must be edited each time it is used to change any of the data-taking parameters (like object names or integration times).

```
filter 1
exptime 30
object 'UGC 12176 @ J'
mgo 5
filter 2
exptime 60
object 'UGC 12176 @ H'
mgo 3
filter 3
exptime 40
object 'UGC 12176 @ K'
mgo 5
end
```

Note that all scripts must end with the END command as shown above.

To use this script, you would type each of these commands into the procedure buffer using the PEDIT command, and then execute it with the RUN command. You can also save it to disk using the WP command, and the either read it in and run it (RP followed by RUN), or it may be executed directly with the CALL command. Examples are given in subsequent sections.

6.2.2 An Image Sequence (Part II)

Suppose now that you want to acquire a 3-filter image sequence for each of a large number of objects. Editing the script in §6.2.1 to change the object name and integration times every time you went to a new object would not only be tedious, it could be dangerous as there are many opportunities to make mistakes.

A more general solution is to write a script that has command-line arguments. You can pass arguments to a script using `PARAMETER` command. By taking the additional step of creating an alias to run the script, you can then treat it as a custom Prospero command. Scripts are your friends.

The script below is in a file named "dojhk.pro" that is stored in the procedure directory. We will execute it via a command alias that uses the CALL command. It also uses a number of other script features as described below. First the script:

```
#
# dojhk - take a JHK image sequence
#
# usage: call dojhk 'object name' Jexp Hexp Kexp
#
# where: 'object name' is the object title in quotes
```

```
#         Jexp, Hexp, Kexp are the integration times for
#           each filter band, in seconds.
#
# Will take 5 H images of Hexp seconds each, 3 J images of Jexp,
# and 5 K images of Kexp, in order, changing the name each time.
#
# 1999 Feb 2 [rwp/osu]
#
parameter string=objid Jexp Hexp Kexp
#
# J-band sequence: 5 images of Jexp seconds each
#
filter 1
exptime %Jexp
string newname '{objid} @ J'
object '{newname}'
printf 'Taking 5 J-band images of %f5.1 sec each...' Jexp
mgo 5
#
# H-band sequence: 3 images of Hexp seconds each
#
filter 2
exptime %Hexp
string newname '{objid} @ H'
object '{newname}'
printf 'Taking 3 H-band images of %f5.1 sec each...' Hexp
mgo 3
#
# K-band sequence: 5 images of Kexp seconds each
#
filter 3
exptime %Kexp
string newname '{objid} @ K'
object '{newname}'
printf 'Taking 5 K-band images of %f5.1 sec each...' Kexp
mgo 5
printf 'Images of {objid} Done'
end
```

To use this, you would write it out into a procedure script file named dobvr.pro with the WP command (Write Procedure):

```
wp dojhk
```

and then define an alias (see §5.1), “dojhk”, to use as a custom Prospero command word:

```
alias dojhk 'call dojhk'
```

To execute this script, you would type the `dobvr` alias with the necessary arguments. For example, to take JHK images of Mrk 35 with integration times/image of 60, 90, and 45 seconds in J, H, and K respectively, you would type:

```
dojhk 'Mrk 35' 60 90 45
```

The result will be five 60-second J-band images, three 90-second H-band images, and five 45-second K-band images of Mrk 35, each labeled appropriately. Note the single quotes (‘) surrounding the object name; these are required to delimit the name string since it contains spaces.

This is only meant as an illustration of how Prospero scripts can be used to automate observing, and is not meant to be an example of good IR imaging practice with OSIRIS.

7 Troubleshooting

This section describes some common problems that observers have encountered while running Prospero. We will add more here once we get enough experience with the system to know what the typical troubles are, and their solutions.

Prospero has crashed (but the workstation hasn't)

Symptoms: Prospero is no longer running and you are looking at a Unix prompt (or a blank workstation screen).

Solution: Restart Prospero in the manner appropriate to your site (i.e., either from an xterm or via a mouse menu). If the data-taking PCs are still running, what you do next depends on what the PCs are actually doing:

1. If PCs are ready but idle (just waiting there), reconnect Prospero using the STARTUP command.
2. If the PCs are acquiring data, let it finish, and then do a STARTUP to reconnect. STARTUP will not allow you to interrupt data acquisition.
3. If one or more of the data-taking PCs have also crashed, restart the PCs before you restart Prospero, following the prescribed local procedure.

One or more PCs have crashed, but Prospero has not

Symptoms: One or more of the data-taking system PCs are no longer responding, and Prospero is timing out.

Solution: Restart the crashed PCs following the local recovery procedures. When everything on the PC-side is back to normal, issue a RESTART followed by a RECOVER from the Prospero keyboard.

The IC complains that it is in MOVIE mode

Symptoms: You attempt to start an integration and the IC reports back an error complaining that you can't start an integration while in MOVIE (continuous readout) mode. Other symptoms are you attempt to query the IC for status, or to change an exposure parameter, and it waits a while and then exits with a timeout error. On looking at the IC monitor display, you see it is in MOVIE mode, or in the process of integrating or reading out.

Solution: In continuous readout mode, the IC locks out most observing commands, since it is engaged in the real-time task of reading out the detector over and over again. To stop MOVIE mode and return to normal, you have these options:

1. Issue the `STOPMOVIE` command from the Prospero keyboard. You will have to wait while the current readout winds down and it does some bookkeeping (*be patient*). If the integration time is set too long (or you get unlucky breaking into the cycle) the command may timeout before you get a reply.
2. If Prospero cannot issue the `STOPMOVIE` command, go to the IC computer keyboard and enter an `ABORT` command. `ESC-]` is also equivalent to `ABORT`. Wait for wind-down and the IC Status to show "IDLE", and then on the Prospero keyboard issue a `RECOVER` to make sure the state flags are all reset.

It is usually a good idea to issue a `RECOVER` if you have any problems getting out of MOVIE mode normally (i.e., via the Prospero keyboard), to make sure the system state tables are updated.

The IC/IE/TC are up, but Prospero doesn't know this.

Symptoms: You can see that a particular data-taking PC is up and running, but Prospero shows red on its status window and refuses to talk to it.

Solution: This is usually a sign that the system status tables have gotten out of synch with each other.

1. Double check that the processes are indeed up and running from the WC status screen.
2. At the Prospero keyboard, issue the `RECOVER` command. This will ask the data-taking PCs to reset their internal communications tables, then update the Prospero status tables.
3. If `RECOVER` fails, try issuing the `STARTUP` command to attempt to fully restart the connection to the data-taking PCs.

This most often happens if someone is doing something on one of the data-taking system PC keyboards at the same time that Prospero wanted their attention, causing a communication conflict.

Prospero can't find my procedure files

Symptoms: Prospero says "cannot read procedure file" when you try to execute the RP or `CALL` commands.

Solution: There are a couple of possibilities:

1. By default, Prospero looks in the currently defined procedure directory for procedure files. Type PRINT DIR and see which directory is the default procedure directory. You can change it using the SETDIR PR=/your/path/here/ command. The default procedure directory is defined at startup by the PR_PRODIR environment variable.
2. If the path is correct, does the procedure file have the .pro file extension? This is the assumed default. Note that all auxiliary files used by Prospero MUST have a "dot"

file extension, even if Unix allows extension-less filenames. Names like "foo.xxx" are allowed, but "foo" all by itself is not.

I closed the status window, and Prospero crashed

Symptoms: You use the mouse buttons to close, quit, or dismiss either of the Prospero status window, LickMongo plotting window, or the TV display window, and Prospero promptly crashes with X broken pipe errors (or some such).

Solution: ***DON'T DO THAT!!!*** Sigh. If you don't want to look at an auxiliary window, or it is getting in your way, turn it into an icon. If you kill the window with the cursor, with current window managers it is the same as killing the parent process. (For you Unix gurus in the audience, it is the same as a KILL signal to the parent. Sweet of it, huh?).

Prospero says the IC/IE/TC/etc. are up, but they don't respond

Symptoms: The Prospero status window shows that the IC, WC, IE, TC, etc. (as appropriate) data-taking PCs are up and running, but when you send commands to one of them, Prospero times out with an error.

Solution: It is possible that one or more of the data-taking PCs have in fact gone offline without telling anyone, and were restarted without telling Prospero about it. The "link" flags on the status window are only updated when Prospero asks for the system status, or if one of the PCs broadcasts changes in its status (which doesn't happen in crashes).

When faced with a process (or processes) that do not respond, issue a `RECOVER` command from the Prospero keyboard. This will ask the WC to update its internal communications table, and then report the current process table status back to Prospero, updating the current display. If that does not solve the problem, it may be necessary to see if the program on the missing process has crashed and needs to be restarted. Refer to the instrument Troubleshooting Guide for details on crash recovery.

The Prospero TV display has locked up

Symptoms: The Prospero image display window is up, but you can neither display new images, nor get cursor feedback (position, zoom, pan, etc.).

Solution: There is a low-level bug in the SunOS OpenWindows. If you start up Prospero and activate the TV display *before* starting a full data-taking session using the STARTUP command, the window manager forgets the TV display exists and refuses to service further action requests. We do not experience this with the generic twm or mwm (Motif) window managers on other systems. We have no identified workaround. The only solution is to restart the Prospero session, making sure that you have successfully connected to data-taking PCs first before attempting to display an image with the TV command.

Prospero is going crazy printing >'s

Symptoms: The command prompt has vanished, and Prospero is rapidly printing line after line of >'s, which are scrolling past at warp speed.

Solution: This is the infamous "Prospero Death Roll". We do not know what causes it, as it seems to be a really weird voodoo pathology of the OpenLook Window manager. It only happens when the *ariel* daemon is killed off prematurely by a PC system crash in the middle of servicing a communications request, but it has been the very devil to reproduce in the lab. If we try to make it happen, we can't.

Which of course does nothing to help you out of it. Some possible solutions:

1. Enter the `QUIT` command. You won't see the words, but it will catch it(usually). When you get the "do you really want to quit" prompt, type "Y" followed by the Return key. With luck, Prospero will terminate and fold up.
2. If that doesn't work, type `Ctrl-Z` to force Prospero into the background, then kill it off with the Unix "`kill -9`" command.

Restart Prospero from the mouse menus (or from an xterm, depending on how your account has been configured to run Prospero).

These seem drastic, but since the *ariel* daemon is dead anyway, it will not do anything bad to the data taking system since the communications link is dead (and, in fact, the WC or IC has probably crashed and need to be rebooted, anyway).

Note: As of late, we haven't seen this problem. Famous last words…

8 Detailed Startup Example

This section gives a detailed, annotated walk-through of a typical Prospero session startup from running the program through connecting to the data-taking PCs. We will do this for a generic instrument setup. Please note that a few details will change depending on the instrument.

Running the Prospero Program

When Prospero is first executed, it will occupy at least two windows on the workstation screen: and xterm for commands and text output, and a custom status window created by Prospero on startup. On startup, Prospero clears the xterm screen, prints a welcome message, and attempts to load and run the default startup procedure script, like this:

```
 Prospero Version 4.3 - (1999 Feb 07 16:25:08)

                       *******************
                       Welcome to Prospero
                       *******************

                    The Ohio State University
                     Department of Astronomy
         Instrument Control and Image Acquisition System
             Cerro Tololo Inter-American Observatory
                      4-m Blanco Telescope

-----------------------------------------------------------------
 To see what's new, type NEWS

1 PR>
```

Note that Prospero begins with the current version number (in this example "4.3"), and the date this version was last updated (UTC date and time). This is followed by the welcome message, sometimes identifying the site, and then a note saying that the user startup procedure script is being executed.

Since the Prospero command prompt appeared in this example without any chatter after loading the startup procedure, you are ready to start issuing commands.

The Prospero STARTUP Process

Once you get the Prospero command prompt, and if the data-taking PCs are running and ready, you next need to connect Prospero to them via the STARTUP command. This will then begin to fill the screen with the various startup chatter, report on the current system state (which instrument is connected, disk drives, log files, etc.), and ask some important questions of the user as it proceeds to verify some important low-level settings. For example:

```
1 PR> startup
 connecting to ariel...
 Performing Instrument-Dependent Setup...
 Loading the OSIRIS instrument configuration...
 Loaded Filter Table ...
 Getting instrument status ...
 The IC and WC disks are synchronized and ready...
 The Caliban & ICIMACS data-transfer disks are synched...
 Image Directory [/data/observer] ?
 Image storage is ENABLED...
 The next file to be written is: test.002
 Is this OK <y|n> [y] ?
 TC Link has been enabled ...
 System clocks synchronized ...

 STARTUP Completed...
2 PR>
```

Let's dissect this screen of stuff and see what is happening.

Summoning ariel

Prospero connects to the data-taking PC computers via a Unix daemon program named *ariel*. Ariel mediates and logs all traffic between Prospero and the PCs, and provides interfaces to other subsystems like telescope controllers and data loggers (e.g., at SAAO ariel is also the interface into the telescope controller emulator).

The first step in starting up an observing session is for Prospero to "summon" ariel and make contact with the data-taking PCs:

```
1 PR> STARTUP
 connecting to ariel...
```

If the ariel daemon is present, you will see it go on with the next steps of the startup process.

If the ariel daemon is absent, you would quickly receive an error message like this:

```
 connecting to ariel...
 Cannot open /tmp/ariel.out...
 Comm Error: No ariel daemon present
*** Warning: the ICIMACS Link is DOWN ***
Suggestions:
  1) Is the WC up and the ariel daemon running?
  2) Are the /tmp/ariel.in or /tmp/ariel.out files absent?
  3) Is /tmp full such that ariel cannot write the log files?
  4) Is ariel sending random garbage?  Check the ariel logs
     for out-of-synch messages; you may need to restart ariel.

STARTUP aborted with errors...
```

The most common solution is #1: the WC is either not running, or if it is, it has not yet connected to the Sparc station and started the ariel daemon. See the WC troubleshooting guide for details on what to do.

If ariel is present, but the data-taking PCs are not responding, then after a wait of 30 seconds or so, the request for info from either the WC or IC will timeout and you will see an error message like this:

```
Error: Requested Operation Timed Out
**ERROR: Could not read the IC or WC status...
Suggestions:
  1) Verify that the IC and WC are up
  2) Check the host tables (HOST command) on
     the IC & WC
  3) Make sure that the WC is in REMOTE mode
  4) Check the ariel log files for garbage or
     out-of-synch messages.

STARTUP aborted with errors...
```

The suggestions printed are to guide you to some possible next steps to take to solve the problem. Refer to the troubleshooting guide (or your telescope operator) for assistance.

Connecting to the data-taking PCs

The data-taking system at present consists of at least 2 PC computers: an "Instrument Computer" or "IC", and the "Working Computer" or "WC". The IC contains the detector array controller hardware and is the primary interface to any instrument control computers. The WC is the computer that makes the Ethernet connection to the Unix workstation and acts as the other end of the connection with the ariel daemon. It also passes all data in FITS format from the IC to the storage disks on the Unix workstation (or to the WC disks in a pinch). It also acts as the central hub of the data-taking PC system, and occasionally provides connections to other processes, like telescope controllers, that are outside the normal data-taking system.

For instrument with computer-controlled mechanisms, a third computer, named the "Instrument Electronics" or "IE" computer, will also be connected to the system. Additional detectors or major subsystems will often have their own dedicated PC computers, and links will be enabled as required.

Most of the startup negotiation between Prospero and the data-taking PCs happens silently, but while it is going on, you should start to see information being written (in white, yellow, green, and red letters) onto the Prospero Status Window. This should let you follow the progress of the startup.

If there is a computer missing or some problem, you would get a warning/error message something like this:

```
IE Link is Disabled...
Suggestions/Possibilities:
  1) Is the IE power on and are the cables
     installed correctly?
  2) If the IE is linked via the head electronics:
     a) is the HE powered on and connected?
     b) is the Sequencer enabled?
```

```
3) Perhaps this instrument does not have an IE?
   (e.g., only an IC/WC for detector work).
```

The IC computer tells Prospero what instrument is connected, and if it has an IE, it attempt to connect up to it and ask for the current instrument status. This info is used to figure out which filters, camera, etc. are in place. If it is absent, you cannot proceed.

Instrument Setup

Once Prospero is connected to the data-taking PCs via ariel, it begins a set of instrument-dependent setup steps. This leads to the next part of the startup chatter:

```
Performing Instrument-Dependent Setup...
Loading the OSIRIS instrument configuration...
Loaded Filter Table ...
Getting instrument status ...
```

This part shows that the instrument has been recognized (it is OSIRIS), and Prospero has loaded the necessary configuration tables. The instrument configuration file is stored in an external file (called osiris.ini) that contains all of the necessary setup info. If for some reason Prospero cannot find this file, you would get an error message and the startup would terminate. If that happens, it means things are pretty seriously messed up, and you should start Prospero over again, following the instructions to the letter (if you *did* that, time to seek outside help).

Image Storage

The next step in the startup is to verify the image storage state, and that the image filenames are valid.

```
The IC and WC disks are synchronized and ready...
The Caliban & ICIMACS data-transfer disks are synched...
```

These lines verify that the data-transfer disks that link (1) the IC and WC computers, and (2) the PCs and the Sparcstation (via the Caliban daemon) are connected and synchronized. If any of these are not synchronized, data cannot flow between the array controller (the IC) and the Sparcstation disks. In this case, you must consult the troubleshooting guide for the correct recovery procedure, as that changes more rapidly than this manual can keep track of.

If the disks are synchronized and the data pipeline is ready to roll, you are then asked to verify the data directory and the filename of the next file to be written:

```
Image Directory [/ua11/iraf/osiris] ?
Image storage is ENABLED...
The next file to be written is: test.002
```

These lines show that all raw images from OSIRIS will be stored to a disk filesystem named "/ua11/iraf/osiris" on the SPARCstation, and that the next file to be written will be named "test.002". You are then asked to verify this:

```
Is this OK <y|n> [y] ?
```

Hitting Return at this query will say “Yes”, accepting the default option (the y enclosed in the []’s). If you want to change the filename, you would enter “n” (the only other choice given inside the <>‘s), and you would be asked for the new filename:

```
Is this OK <y|n> [y] ? n
Enter a new filename (xxxxxxxx.nnn) [test.002] ? 990210.001
```

In this case, the file “`990210.001`” will be the next file written by the system when data is actually taken.

The data-taking system uses the file extension as a way of numbering the data files, so the extension must be a number, in this case “001” was given. Subsequent files will be named `990210.002, 990210.003`, and so on, up to `990210.999` for a maximum of 1000 files (`990210.000` is also valid, it comes before 001, but is rarely used). See §4.4.2 for a discussion of filename conventions.

Mop Up

After establishing the image storage setup, there will be a few more leftover startup procedures to go before ending. For example, in this example we see:

```
TC Link has been enabled ...
System clocks synchronized ...
```

Which tells you that the telescope controller (TC) is enabled and ready. In this case, it is a low-level TCS interface running on the WC. The second line tells you that the system clocks have all been synchronized with each other (the standard is the Unix time clock read by the ariel daemon).

Finally, at long last, you should see:

```
 STARTUP Completed...
2 PR>
```

The prompt is now “PR>“. This tells you that the instrument is connected and you are ready to begin taking data.