NIRSPEC

UCLA Astrophysics Program **U.C. Berkeley** **W.M. Keck Observatory**

James Larkin **January 30, 1998**

NIRSPEC Software Programming Note 25.00
Starting IDL programs as Stand Alone processes

1. Introduction

There are two NIRSPEC software packages (the **EFS** and **Quicklook**) that are IDL applications and which therefore require that an IDL process is started to run them. An individual IDL process can only control one widget manager, so there are two ways to have two or more IDL applications going. One procedure would be to write a master IDL procedure which opens all of the windows widgets for all applications and which has a master event procedure that processes key and mouse events and directs the appropriate responce from the correct application. This would be very clumsy at this stage, and would greatly limit the flexibility and modularity of the NIRSPEC software (i.e. it does not make sense to tie the **EFS** and **Quicklook** to be combined in any manner). The second option is to use a different IDL procedure for each application and launch the programs fully independently. It is this second option that we have selected and that is described in this programming note.

2. Overview

In each of the two cases (**EFS** and **Quicklook**) the programs consist of a series of IDL procedures that open XWindows Widgets from within IDL. To run with other applications, however, both procedures must be started from the general csh script that is used to start all of the NIRSPEC client software (**run_nirspec_client**). To accomplish this, IDL must be started in server mode which allows other programs to use **RPC** communication to issue IDL commands. A program is then written in C which issues **RPC** calls to the IDL server to begin the appropriate IDL program (i.e. EFS or **Quicklook**). When the application is ended, a second C program is used to issue an IDL exit command through **RPC**. To coordinate these C programs, a csh script is written for each IDL application. The master **NIRSPEC** client start procedure calls each of these coordinating csh scripts as background procedures.

3. CSH script to coordinate RPC communication with IDL servers

The example below is for the **EFS** but an almost identicle script works for the **Quicklook** as well. **CSH** code called **run_efs**:

```
#!/bin/csh -f
# Stand alone script to launch the EFS package within IDL
source /kroot/kui/qlook/idl_setup
xrdb -merge /crab/home1/nirspec/.Xdefaults
xterm -iconic -e idl -server=20500000 &
sleep 10
/kroot/kui/efsgui/rpc/nirspec_efs
```

```
/kroot/kui/efsgui/rpc/exitidl
```

The first line: **#!/bin/csh -f** is used to specify the type of script. By sourcing the **idl_setup** routine, IDL is setup with the appropriate paths for NIRSPEC applications. The **xrdb** command makes sure xwindows definitions are the same for all NIRSPEC programs. The **xterm** command starts an IDL program in server mode within an xterm window that is iconified at the bottom of the screen. The flag **-server=20500000** specifies the process id number for the IDL server. The same id number must be used by any process that sends commands to the IDL server. **Sleep 10** waits 10 seconds to make sure the IDL program is properly started, and then the C executable **nirspec_efs** is started. This executable described next issues the RPC command to start the **EFS** procedure. The shell will pause here until the user exits the **EFS**, at which time the next line is executed to run the C executable **exitidl** that kills the IDL server.

4. NIRSPEC_EFS C Code

This section describes the small C program that communicates with the IDL server to start the **nirspec_efs** procedures. The program is called **nirspec_efs.c** which is compiled to be just **nirspec_efs** in the **/kroot/kui/efsgui/rpc** directory. The outline of the c code is:

```
/* nirspec_efs.c */
/* include files */
***
main(c,v)
/* Define variables. */
***
/* Connect to the client */
***
if ((client=register_idl_client(0x20500000,hostname,&timeout))==NULL)
      {printf("Can't find serer.\n");
      exit(1);}
/* Send the commands to the server */
if(send_idl_command(client,".r nirspec_efs.pro")!=1)
      printf("Error encountered sending command!\n");
if(send_idl_command(client,"nirspec_efs")!=1)
      printf("Error encountered sending command!\n");
/* Disconnect from server. */
unregister_idl_client(client);
```

Most of the code is fairly self-explanatory, but it is important to remember to match the id number here with the one used to start the IDL server. The first IDL command compiles the efs procedures, and the second one starts the program.

The **exitidl.c** code is virtually identical to the **nirspec_efs.c** except the IDL command sent to the server is **"exit"** instead of **"nirspec_efs"**. The reason to split the tasks of running the **EFS** and exiting IDL is that the **run_efs** csh script will naturally wait for the **EFS** code to end before going to the next step that exits IDL.

5. Putting it all together

Once the C executable is created and the **run_efs** script are complete, it is possible to just execute the **run_efs** script to start the **EFS** and properly exit IDL when you're done. To start all of the client software, the **run_nirspec_client** shell script just needs lines like the following:

```
# run EFS program
if (!($noefs)) then
      /kroot/kui/xnirspec/run_efs &
end if
```

By adding the &, each IDL procedure is started in the background and the script can continue on to start the remaining software. The conditional just checks to see if the flag **-noefs** is used by the user.