Data Reduction Helpdesk First Responder Guidelines

Kathleen Labrie

Science Users Support Department

V1.1 – 12 March 2018

Revision History

V1.0 – 16 January 2018 Kathleen Labrie
V1.1 – 12 March 2018 Kathleen Labrie

Document ID: DPSG-USER-101_DRHDFirstResponderGuideline**s**

Document Purpose

This document aims to help the Tier 1 assignee respond to the data reduction tickets in a manner that will gather the necessary information for understanding and debugging the problem reported by the users in a timely manner.

Intended Audience

This document is intended primarily for Tier 1 Data Reduction Helpdesk assignee, who are often NGO members. Members of the Science Users Support Department new to handling data reduction tickets should also read this.

Table of Contents

1. Introduction

When a user file a Helpdesk ticket reporting a problem or asking a question about data reduction, there are very specific steps that should be taken to ensure a timely resolution of the issue. As a first responder, your responsibility is primarily to gather information. Many problems will require a Gemini data reduction software developer to troubleshoot the issue by looking and/or fixing the code. In order to do that, there is a set of critical information that is needed. This is where the first responder role, Tier 1 in the current Helpdesk system, come in. A good job at that level can dramatically speed up the resolution of the issue, which is of course something the user will appreciate.

Below we describe the specific steps that can be taken by the Data Reduction Helpdesk First Responder, from setting up your computer to the information gathering steps that are needed.

2. Preparations – Assembling your First-aid Kit

Before addressing any data reduction ticket, the first responder needs to prepare his or her computing environment. Troubleshooting software most often requires running the software, therefore one needs to be prepared.

2.1 Choosing a platform

Gemini offers support for both Linux and Mac OS X operating systems. In an ideal situation, you would have both a Linux machine and a Mac OS X machine fully set up, ready to go. This would allow troubleshooting on the same type of machine the user is using. However, it is understood that this is probably not very realistic. It's okay. If you can do it, otherwise pick either Linux or Mac OS X, they are similar enough and behave the same way in most cases anyway.

For Linux, the platforms we officially support are CentOS 5, 6, and 7. Support of CentOS 5 will soon be dropped. Ideally, you would be using one of those, but any recent Linux distribution will do as first responder.

For Mac OS X, the versions we officially support are 10.8 and above. It is expected that we will soon drop support of anything below 10.10.

At the end of the day, you are most likely already using a computer with a compatible platform. So the guidelines on the platform are just that, guidelines.

Summary

Minimum requirement, one of these:

- Linux – CentOS 5 and up or equivalent
- Mac OS X – 10.8 and up

2.2 AstroConda

A more critical preparation is the installation of the appropriate software. Anaconda is a software distribution system. AstroConda is a suite of software used in astronomy that can be installed with the Anaconda system. It is mostly gear towards Python support but Gemini has included in AstroConda the necessary software to install IRAF, Gemini IRAF and all its dependencies, and also a large collection of other IRAF packages.

The specific version of Anaconda, or the versions of the various software installed through AstroConda will change continuously. As long as your installation is no more than a year old, either through fresh install or through an update, it should be sufficient for first responder duties. That is unless a specific need to update or upgrade has been identified and communicated to the NGO and the public.
Instructions for a fresh install of AstroConda (and incidentally Gemini IRAF), see the "New Installation of Gemini IRAF from AstroConda" section on this Gemini webpage:

http://www.gemini.edu/node/11823

Summary

- Install AstroConda following the instructions here: http://www.gemini.edu/node/11823
- Make sure your installation is no more than one year old.

2.3 IRAF and Gemini IRAF

The most critical software is obviously IRAF and the Gemini IRAF package. Versions need to match that of the user.

2.3.1 IRAF

The IRAF version distributed with AstroConda is very unlikely to change anytime soon. The only reason it would change is if a critical bug is discovered and fixed. So, when it comes to IRAF, just make sure you are indeed using the IRAF installed by AstroConda. A way to confirm this can be by querying the system about the path to pyraf. For example:

In tcsh:

```
% which pyraf
/Users/klabrie/anaconda/envs/geminiconda/bin/pyraf
```

In bash:

```
% type pyraf
pyraf is /Users/klabrie/anaconda/envs/geminiconda/bin/pyraf
```

The path in bold is a good indication that you are indeed using the AstroConda distribution of IRAF and pyraf. (On truly messed up installations this might not be 100% full proof, but it is a good basic check.)

2.3.2 Gemini IRAF

When troubleshooting, the Gemini IRAF version you use **must match** that of the users. That is not a guideline, that is an **absolute necessity**. If the user is using an ancient version of Gemini IRAF, the first step is to ask them to update and see if that solves their problem. It often does.

A good test environment would always have the latest version of Gemini IRAF installed. An excellent test environment would also have the previous version installed too, especially when a new released just came out as that's when you are most likely to receive tickets from people using the version on either side of the release. With anaconda, it is actually really simple to install two version of Gemini IRAF and switch from one to the other by switching anaconda "environments". If you are interested in setting more than one anaconda environment and are not sure how, contact the Science Users Support Department, we will help you out.

If you have installed a fresh AstroConda, you already have the latest Gemini IRAF package. If you have already AstroConda installed and a new version of Gemini IRAF is released, you can update by doing the following (assuming your environment is named "geminiconda"):

```
% source activate geminiconda
% conda update gemini
```

Summary

- Ensure that you using the AstroConda version of IRAF
- Always have the latest Gemini IRAF installed.
- If you can, have the previous version of Gemini IRAF installed in another anaconda environment.

3. Action 1 – Assembling the Debug Kit

In order to help the user, we need to understand **what they are using** and **what they are doing**. The first objective is to **reproduce the problem**. Being able to reproduce the problem is half the battle when it comes to debugging software or helping a user with usage issues.

In order to do that, we need information, a lot of information. All this information needs to be obtained from the user. Hopefully, some of the information we need is already included in the ticket, but we still need the rest of it. The ticket assignee needs to engage the user and collect all that info before moving on.

While some of information might not mean anything to you, do ask for it anyway. Your role here is to collect the information. If it turns that you need to elevate the ticket to Gemini, that information will be useful to us as we dig deeper into the issue. By collecting everything as the first step, when the ticket is elevated we can start our work right away, without having to wait for additional inputs from the user.

So, let's discuss what is needed to build a complete "Debug Kit".

3.1 Overview of information to get from the users

A "Debug Kit" will contain the information and data listed below. Here we just have a sort of checklist. In the following subsections, each will be discussed and described in more details.

Checklist

- ❒ Operating system (eg. Linux CentOS 6, Mac OS X 10.11)
- ❒ Version of IRAF and external packages
- ❒ Version of PyRAF
- ❒ Version of Gemini IRAF
- ❒ Minimum set of data to reproduce the problem, including necessary calibration files
- ❒ Exact command being issued
- ❒ Full list of input parameters (from the `lpar` command)
- ❒ Sometimes, the lpar outputs of the previous steps can be needed.

3.2 User's system information

3.2.1 Operating System

The user should be able to tell you rather easily whether they are using Linux or Mac OS X. But we might need more details then that.

Ask the user for the output of the following commands:

Linux users:

```
% uname –a
% lsb_release –a
% cat /etc/*release
```

Mac OS X users:

```
% uname -a
% sw_vers
```

3.2.2 Software Versions

If all the Gemini IRAF and dependencies software was installed with anaconda, it is very easy to get all the version numbers we need in just a couple commands.

Ask the user for the output of the following commands:

```
% conda list iraf
% conda list -f pyraf
% conda list –f python
% conda list –f numpy
% conda list –f matplotlib
```

Pay particular attention to the version of `iraf`, `iraf.gemini`, and `pyraf`. At the time of this writing `iraf` should be version 2.16.UR and `pyraf` should be 2.1.11. The version of `iraf.gemini` being likely to change more often, please just refer to the Gemini website to check the latest version number.

Another set of information on versions is obtained from within IRAF. Ask the user to run the install_check.cl script and share the output with you:

```
% cd iraf
% pyraf
--> gemini
--> task $install_check=gemini$install_check.cl
--> install_check
```

3.3 Minimum data set to reproduce the problem

The biggest breakthrough when troubleshooting a software issue is often to be able to reproduce the problem. Therefore, to do that, we need the data the user is using when the problem occurs. Of course, we don't need (and don't want!) gigabytes of data. Ask the user to provide you with the minimum set of files needed to reproduce the problem. This is likely to include processed calibration files.

Let's run through an example. The user is reporting a crash in gireduce when running it on a list of science frames. The gireduce command in the ticket looks like this:

```
--> gireduce @mysci.lis bias="N20120202S0001_bias.fits" \
flat1="N20120202S0010_flat.fits"
```

If the problem occurs on the first frame, and the `mysci.lis` contains 10 frames, we do not need all 10 frames, we just need the first one. We will also need the processed bias `N20120202S0001_bias.fits` and the processed flat `N20120202S0010_flat.fits`. For this example, it might be a good idea to also ask for `mysci.lis` file to check it for random characters.

It is very unlikely that the data can be sent by email. The user will have to upload the data somewhere where you can get to it. Ideally, it should also be a location where the Gemini staff

can also get to it if the ticket is elevated. For privacy reason, the location and access information is normally shared by email rather posted in the ticket.

Typical data sharing tools include a good old FTP directory, some users have used Dropbox, or other cloud solutions, like Google Drive. If neither the user nor you can set up a file sharing solution, contact the Science Users Support Department, we will see how we can help.

3.4 Exact commands issued by the user

Now you need to know the exact command issued by the user that leads to the problem. Do not assume anything. Ask the user to send you exactly what is being typed.

In some cases, it might become necessary to also request the commands that were issued before, for example to understand how the input to task was produced. Those situations are normally elevated to Gemini so you should not have to worry about that.

3.5 Full list of input parameters

IRAF uses parameter caches to store defaults for input parameters. This means that the command issued by the user does NOT contain all the information. Maybe the user is simply using the defaults, but maybe some of those parameters have been changed before the command was issues. Sometimes the user does not even realize that this has been done. Again do not assume anything, get the information.

To get the full list of values for the input parameters, the ones that will be used by the command if not overridden on the command line, ask the user for the output of:

```
--> lpar name_of_the_task
```

Using the example from Section 3.3, the user would run `lpar gireduce`.

---

4. Action 2 – Troubleshooting

Once the Debug Kit is assembled it is time to try running the problematic command and see how it behaves on your system. Remember, as a First Responder, your main duty is to gather information, you are not expected to solve the problem. Of course, if you can, great!

Download the data, select your environment that best matches the user's and run it. Use exactly the same input data, the same input parameters, and the type the same command.

What to look for:

- Does it work for me?
- Can I reproduce the problem?
- Do I know what's going on? Have I seen this before?

If you have a suggestion for the user, go ahead and share it. If not, then your troubleshooting is done. Now it's time to collect all the information, including the results of your troubleshooting and elevate the ticket (Section 5).

5. Action 3 – Elevating to Next Tier

The important task of information gathering is completed. It is quite likely that the ticket needs to be elevated to the next tier. Problems with data reduction software often requires intimate knowledge of the code, it is perfectly okay to elevate a ticket once the information has been collected from the users and once you have tried it yourself on your system. We cannot stress enough how valuable that information is.

In your ticket elevation:

- Communicate to the new assignee where to find the data that the user uploaded. (by email)
- Make sure that all the "Debug Kit" information is in the ticket.
- Include the results of **your** troubleshooting. E.g. were you able to reproduce the problem?
- Include **your** system information, Section 3.2. (It is likely to be slightly different from the user's).

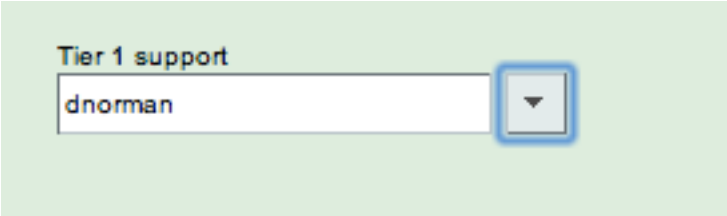
6. Using the Helpdesk System

The Gemini Helpdesk uses the Remedy Helpdesk System. Here are a few things to note about how this software works and how we use it.

6.1 Categories and assignments

- Beware! Changing the category for the ticket does NOT re-assign the ticket to the people assigned to that other category. The assignee will not change.

- You can only reassign to whoever shows up in the pop up menu identified by the down arrow.



6.2 Status definitions and resolution

6.2.1 Status Definitions

New	No one has looked at it yet.
Assigned	Someone is assigned to the ticket but no work has been done yet.
Work in Progress	Someone is actively working on this.
Resolved	A solution has been found (and the user agrees).
Closed	Solution or no solution, when thre is nothing more to do.

6.2.2 Closing a ticket

Solution message when the user has gone quiet and you are not sure if the user got what they needed

"I have not heard back from the user in 3 weeks. Setting the ticket to Resolved. Please revive by posting a reply if you still need help with this ticket."

6.2.3 Reopening a ticket

Only Gemini staff can change status of Resolved or Closed ticket.

Unfortunately, only Gemini staff can change the status of a Resolved or Closed ticket. However, an external user or NGO partner can still write to the ticket. This will trigger the normal notice email. Therefore, "re-opening" a ticket should be understood as adding a new message to a ticket. If you wish a revived ticket to have its status change, contact a Gemini staff member.

7. Debug Kit Reference Sheet

All the details are discussed above in Section 3.2, here we simply provide a quick look reference sheet of what to ask to the user.

System Information:

If Linux:

```
% uname –a
% lsb_release –a
% cat /etc/*release
```

If Mac OS X:

```
% uname -a
% sw_vers
```

Software Versions:

```
% conda list iraf
% conda list -f pyraf
% conda list –f python
% conda list –f numpy
% conda list –f matplotlib

% cd iraf
% pyraf
--> gemini
--> task $install_check=gemini$install_check.cl
--> install_check
```

Data:

- The minimum set of files to reproduce the problem.
- Upload that data to a location where the support person can download them.

Command that triggers the problem:

- The exact command to use to reproduce the problem with the data uploaded.

Input Parameters:

- All the input parameters for the task causing trouble

```
--> lpar name_of_the_task
```

8. Detailed Revision History

v1.0 16 January, 2018 Kathleen Labrie

Initial revision building on slides from a couple years ago and updated to use current software, eg. Astroconda instead of Ureka.