NOAO SDM Document PL027

The NHPPS Simple, Light-Weight Message Protocol

F. Valdes[1], F. Pierfederici[1], M. Miller[1], D. Scott[1]

National Optical Astronomy Observatories
Science Data Management

V1.0: January, 2012

[1]NOAO Data Products Program, P.O. Box 26732, Tucson, AZ 85732

Abstract

The NOAO High-Performance Pipeline System (NHPPS) uses a common simple message protocol for communication between various clients and servers. The message protocol is designed to be very simple and light-weight for ease of implementation in various languages. The protocol is simple ”KEYWORD = VALUE” text which is currently transported by standard sockets in the NHPPS applications where it is used. This document provides the specifics for the message protocol and serves as an ”interface control document” for those applications.

Keywords: NHPPS, messaging, ICD

1 Description

NOAO High-Performance Pipeline System (NHPPS) clients and servers communicate via a simple, light-weight, text-based message protocol. The messages are transported via standard sockets. The protocol and transport makes writing and capturing messages in applications simple in any language. It is this simplicity, and the light-weight nature of the applications, that justifies a custom NHPPS protocol.

NHPPS includes API classes for python applications. One class is for composing and decoding the message format as described below. Another class, layered on the format, provides the standard socket-server functionality. This is for messages that use the reserved ”command” parameter to specify an interpreter method for the message; i.e. like a remote procedure call. Each command value is then a message type where other elements of the message provide parameters specific to that type.

So while the message protocol does not require use of the command parameter, all current uses of the protocol in NHPPS applications consist of such messages to define the type and interpreter to use. Note also that the message protocol need not be transported only through sockets, though that is also currently the case for all applications.

There is nothing special about the message protocol and other messaging formats and systems could be used. The purpose of this document is to describe the current system in use in NHPPS applications. We also note that NHPPS does make use of a more complex remote procedure call system, PYRO, within the python-based pipeline managers for the more demanding distributed pipeline processing framework.

1.1 Format

- Messages consist of lines of text in ”KEYWORD = VALUE” form.
- Messages are terminated by a line with just EOF (case-insensitive).
- Lines are delimited by newlines.
- Blank space around the equal sign is optional.
- Anything after the value is ignored.
- Blank lines and lines beginning with ’#’ are ignored.

1.2 Keywords

- Keywords may not contain whitespace, equal, or quotes.
- Keywords may be arbitrarily long.
- Whitespace before the keyword is allowed.
- Keywords are case-insensitive.

- Reserved keywords are:

  COMMAND: identifies the message interpreter
  HOSTNAME: identifies the client host
  STATUS: return code from call (numeric and string description)
  LABEL: user-defined field

1.3 Values

- Values can be arbitrarily long.
- Values with whitespace must be quoted.
- Values of ”” or ’’ (a zero-length string) are allowed.
- Opening and closing quotes must be of the same type.
- A backslash be used to escape a quote that matches an opening quote, a newline or a ’\’ character in a string.
- Quoted strings may used within a quote-delimited value provided the type of quote (double or single) differs from the value’s opening quote or by using a backslash:
  KEY = ’A B C ”DEF” \’FOO’

2 Examples

Here we give one quick example. The GETCAL client requests the most appropriate calibration from a Calibration Manager server. The request requires providing various parameters about the image to be calibrated such as the time of the exposure, the specific detector, the filter, etc.

This example illustrates a couple of things mentioned earlier. First the multi-language aspect where the GETCAL client is written in IRAF/SPP and the Calibration Manager is written in python and makes use of the python library classes summarized above. The other is that the COMMAND parameter is used by the python library class to call a method mgr_getcal (where the mgr_ is automatically added) in the Calibration Manager application. Another interesting point is that to generate this example the socket connection specified in the GETCAL client was replaced by STDOUT (which the client interprets in the obvious way) to simple print the message to the terminal. This is useful for debugging the client.

Figure 1: Message from IRAF GETCAL client to Python Calibration Manager

```
COMMAND = getcal
CLASS = 'dflat'
MJD = '55931.01234'
DMJD = 3.
DETECTOR = 'like mosaic1'
IMAGEID = 'ccd1'
FILTER = 'V'
QUALITY = 2.
MATCH = 'like 1 1'
VALUE = ''
EOF
```