# Documentation Requirements for SDSS Observing Software

## Objective

The goal of this policy is to ensure adequate documentation at APO for observing-related software that explains:

1) what each command/GUI element does;
2) the function and units of each command and its derivative options;
3) what the output means.

Note: this specifically does not include details about how the code works internally nor details on commands not intended for, or expected useful, in general/emergency use. (Developers are of course welcome and encouraged to provide the project with such documentation, which will undoubtedly not go wasted.)

When new software is tested, the observers have an additional set of objectives and require additional documentation which details:

1) the context of the command in observing procedures;
2) suggested tests of the command;
3) how to judge code failure vs. success; and
4) what other commands are affected by the new command.

The SDSS observers are developing a plan to automatically assemble this documentation into a readily accessible, searchable format. For example, the *OP programs are arranged in such a way that WWW pages can be automatically generated and indexed through the developer-supplied help strings. Guidelines and examples on how to write policy-compliant help strings in *OP have been posted at: *http://sdsshost/sdssProcedures/text/iopHelp.txt*. For programs where this model does not work, the documentation should be delivered in some form that can be put (and indexed) on the web, either in HTML or simple text. (Note that Microsoft Word is NOT an allowable format; TeX is negotiable.)

It is noted that a significant amount of code, which already exists at APO and is used in SDSS observing operations, does not currently meet the requirements of this policy. Since a significant amount of effort will be required to bring all code into compliance with this policy, the initial intent is to apply the requirements of this policy to new code, or modifications to existing code. Over time, the goal will bring all code into compliance, but in the short term, the goal is to ensure that new code, or modifications to existing code, is adequately documented.

## Specific Requirements

As already stated, the documentation requirements actually have two components: what is required for successful testing of new work, and what is required for standard-use documentation.

The following is required for each new command/GUI in general.

1) For command-line commands, a full and complete help string is expected. The help string should be brief, but should not leave out essential information in the interest of brevity. The help string should contain a full summary of what the command does as well as a description of all command line arguments and what they do. Again, needed details should not be sacrificed for brevity. Specifically,

units should be included for all command input parameters and output. For *OP commands, the ftcl help facility is expected to report the command's help information.

A reference/user's manual containing this information (provided it is in one of the WWW-indexable forms described above), in lieu of command-line help, is an acceptable substitute (both are even better).

2) Optionally, for command-line commands, the developer may supply some "extended" help messages that will form more of a User's guide. For *OP commands, refer to the on-line site referenced above. The required information described above is meant as reference material; the extended information is meant to partially bridge the gap to a User's Manual. Things that may be put in the "extended" message include Examples, Known Issues, See Also, and Source Code (the file which contains the source for this command). It is expected that the software users will expand the "extended" help as they see fit, but that the developer should update the reference material. Also, developers are encouraged to talk to the observers about modifying the observers' standard procedures (see: *http://sdsshost.apo.nmsu.edu*) to include new commands.

3) The developer should provide the observers with a way to search a command based on a keyword in its name/ description. For *OP commands, this ability is provided automatically by the examples found at the website listed above: both items of help information will be automatically compiled into an indexed, searchable, set of web pages at the time of *OP-building. If the developer wishes to edit the contents of these automatically produced web pages, do NOT edit the web page itself – change the source code where the help strings are defined instead.

4) For GUIs, an explanation of what each button/interactive element does is required, as well as a description of what the GUI itself is to be used for, and how to invoke it. This documentation should be available directly from the GUI and could take at least two different forms:
    1) A straight manual invoked from a "Help" button much like the mcpMenu uses.
    2) Context-sensitive help for each GUI element that is popped up somehow in an intuitive way. Examples of this technique are Microsoft Window's pop-up balloon messages and the Watcher's pop-up indexed Netscape pages.

5) For any substantially new program or GUI, some sort of manual is required that describes in more detail (than the help strings can provide) what the program/GUI does and how to use it effectively.

Problem Reports (PRs) will be filed against software that does not meet these documentation criteria.

When a request is made to test software during a shakedown/shakeup, the following additional information must be provided:

1) The command to be tested.
2) What the command does and what the observers should base pass/fail criteria on. That is, how is it determined whether the command was successful or not?
3) A description of how the command is intended to be used, procedurally.
4) A list of other commands affected by this new command (commands that this command will replace, modify, etc.).
5) The developers contact information (phone #/email address as appropriate) and availability time during the shake period.

If the command to be tested will be put into daily use after the shake (i.e., the observers are not testing part of an in-progress command), the developer must provide the 5 documentation items listed above.

It is preferred that the additional shake information be sent by email to the Lead Observer (currently sjnk@apo.nmsu.edu), but it may also be included in the GNATS entry when the status is changed to needstest and the assignment is changed to the Lead Observer (currently sjnk). If all this information is not included, the developer risks: a) the code not getting tested and/or b) a late-night phone call asking for clarification.