IDLdoc 3.3 Reference Guide

Michael Galloy

Abstract This reference guide simply lists options available for running IDLdoc and documenting code. See the tutorial for a more friendly introduction to using IDLdoc.

IDLDOC routine keywords

There are quite a few keywords to IDLdoc to set various specifications for the output. Also see the "Customizing Output" section for using templates for customized output.

Table 1 – Keywords for the *IDLDOC* routine

Keyword	**Description**
ASSISTANT	obsolete; no longer used
BROWSE_ROUTINES	obsolete; no longer used
CHARSET	set to the character set to be used for the output, default is "utf-8"
COLOR_OUTPUTLOG	set to color output log messages, i.e., warning messages are displayed in red; IDLdoc will attempt to detect whether it is running from a terminal capable of displaying color text, but this keyword can force IDLdoc to attempt color display
COMMENT_STYLE	output format for comments ("html", "rst", or "latex"); default is "html"
COMPLEXITY_CUTOFFS	McCabe complexity to exceed for a warning or flagged; default is *[10, 20]*
DEBUG	set to allow crashes with a stack trace instead of the default simple message
EMBED	embed CSS stylesheet instead of linking to it (useful for documentation where individual pages must stand by themselves)
ERROR	set to a named variable to return the error state of the IDLdoc call; 0 indicates no error, anything else is an error
FOOTER	filename of file to insert into the bottom of each page of docs

... continued on next page

Table 1 – Keywords for the *IDLDOC* routine (... continued)

Keyword	**Description**
FORMAT_STYLE	style to use to parse file and routine comments (“idl”, “idldoc”, “verbatim”, or “rst”); default is “idldoc”
HELP	set to print out the syntax of an IDLdoc call
LOG_FILE	if present, send messages to this filename instead of *stdout*
MARKUP_STYLE	markup used in comments (“rst” or “verbatim”); default is “verbatim” unless *FORMAT_STYLE* is set to “rst”, in which case, the default is “rst”
N_WARNINGS	set to a named variable to return the number of warnings for the IDLdoc run
NONAVBAR	set to not display the navbar
NOSOURCE	set to not put source code into output
OUTPUT	directory to place output; if not present, output will be placed in the *ROOT* directory
OVERVIEW	filename of overview text and directory information
PREFORMAT	obsolete; no longer used
QUIET	if set, don’t print info messages, only print warnings and errors
ROOT	root of directory hierarchy to document; this is the only required keyword
SILENT	if set, don’t print any messages
ROUTINE_LINE_CUTOFFS	number of lines in a routine before warning or flagged; default is *[75, 150]*
SOURCE_LINK	by default, IDLdoc copies the source code into the output; if this keyword is set to 1 (relative link) or 2 (absolute link), then the output documentation will point to the ROOT location of the original source code
STATISTICS	set to generate complexity statistics for routines
SUBTITLE	subtitle for docs
TEMPLATE_PREFIX	prefix for template’s names
TEMPLATE_LOCATION	set to directory to find templates in
TITLE	title of docs
USER	set to generate user-level docs (private parameters, files are not shown); the default is developer-level docs showing files and parameters
USE_LATEX	set to use MathJax to automatically typeset any LaTeX style equations in comments

... continued on next page

Table 1 – Keywords for the *IDLDOC* routine (... continued)

Keyword	Description
VERSION	set to print out the version of IDLdoc

Format styles

rst format style

The following tags are available in file comments, i.e., comment headers not immediately preceeding/following a routine header.

Table 2 – rst format style file tags

Tag name	Arguments	Attributes	Description
:Author:	comments	none	specifies the author of the file
:Copyright:	comments	none	specifies the copyright information for the file
:Examples:	comments	none	specifies examples of usage
:Hidden:	none	none	if present, indicates the file is not to be shown in the documentation
:History:	comments	none	lists the history for the file
:Private:	none	none	if present, indicates the file should not be shown in user-level documentation (set with the *USER* keyword to IDLdoc)
:Properties:	property name, comments	none	describes properties of a class, i.e., a keyword to *getProperty*, *setProperty*, or *init*
:Version:	comments	none	specifies the version of the file

The following tags are available for comments immediately preceding or following a routine header.

Table 3 – rst format style routine tags

Tag name	Arguments	Attributes	Description
:Abstract:	none	none	if present, indicates the method is not implemented and present only to specify the interface to subclasses' implementations

... continued on next page

Table 3 – rst format style routine tags (... continued)

Tag name	Arguments	Attributes	Description
:Author:	comments	none	specifies the author of the routine
:Bugs:	comments	none	specifies any issues found in the routine
:Categories:	list	none	specifies a comma-separated list of category names
:Copyright:	comments	none	specifies the copyright for the routine
:Customer_id:	comments	none	specifies a customer ID for the routine
:Description:	comments	none	a tag for the standard comments for a routine; will be appended to standard comments if both are present
:Examples:	comments	none	specifies examples of using the routine
:Fields:	fields	none	specifies the names of the field followed by a description of the field
:File_comments:	comments	none	equivalent to the main section in file-level comments
:Hidden:	none	none	if present, indicate the routine should not be shown in the documentation
:Hidden_file:	none	none	if present, indicates the file containing this routine should not be shown in the documentation
:History:	comments	none	specifies the history of the routine
:Inherits:	none	none	not used
:Keywords:	params	see below	documents keywords of the routine
:Obsolete:	none	none	if present, indicates the routine is obsolete
:Params:	params	see below	documents positional parameters of the routine
:Post:	comments	none	specifies any post-conditions of the routine
:Pre:	comments	none	specifies any pre-conditions of the routine
:Private:	none	none	if present, indicates the routine should not be shown in user-level documentation (set with the *USER* keyword to IDLdoc)
:Private_file:	comments	none	if present, indicates the file containing this routine should not shown in user-level documentation (set with the *USER* keyword to IDLdoc)

... continued on next page

Table 3 – rst format style routine tags (... continued)

Tag name	Arguments	Attributes	Description
:Requires:	comments	none	specifies the IDL version of the routine; IDLdoc finds the routines requiring the highest IDL version and reports them on the warnings page
:Returns:	comments	none	specifies the return value of the function
:Todo:	comments	none	specifies any todo items left for the routine
:Uses:	comments	none	specifies any other routines, classes, etc. needed by the routine
:Version:	comments	none	specifies the version of the routine

The *keyword* and *param* tags above accept attributes. The available attributes are shown below.

Table 4 – rst format style tag attributes

Attribute name	Syntax	Description
in	`in`	indicates the parameter is an input
out	`out`	indicates the parameter is an output
optional	`optional`	indicates argument is optional
private	`private`	indicates argument is not shown if IDLdoc is run in user mode (*USER* keyword to IDLdoc is set)
hidden	`hidden`	indicates the argument is not to be shown
required	`required`	indicates argument is required
type	`type=comments`	IDL data type of the argument
default	`default=comments`	default value of the argument

The tags available in an overview file describe the entire library. There are a few tags shared with the file tags and the additional *:Dirs:* tag which provides a simple table of contents for the directories in the library.

Table 5 – rst format style tags for overview files

Tag name	Arguments	Attributes	Description
:Author:	comments	none	specifies the author of the library
:Copyright:	comments	none	specifies the copyright for the library

... continued on next page

Table 5 – rst format style tags for overview files (... continued)

Tag name	Arguments	Attributes	Description
:Dirs:	dirs	none	lists directories in the library along with a description for each; excepts `private` and `hidden` attributes in the same manner as the *:Params:* and *:Keywords:* tags for routines
:History:	comments	none	specifies the history of the library
:Version:	comments	none	specifies the version of the library

A file named *.idldoc* placed in a directory is a "directory overview" file. It can contain properties of the entire directory, but does not affect subdirectories. Directory overview files also have a few tags shared with file tags.

Table 6 – rst format style tags for directory overview files

Tag name	Arguments	Attributes	Description
:Author:	comments	none	specifies the author of the files in the directory
:Copyright:	comments	none	specifies the copyright for the files in the directory
:Hidden:	none	none	if present, indicate the routine should not be shown in the documentation
:History:	comments	none	specifies the history of the library
:Private:	none	none	if present, indicates the directory should not be shown in user-level documentation (set with the *USER* keyword to IDLdoc)

IDLdoc format style

The following tags are available in file comments, i.e. comment headers not immediately preceeding/following a routine header.

... continued on next page

Table 7 – IDLdoc format style file tags (... continued)

Tag name	Arguments	Attributes	Description

Table 7 – IDLdoc format style file tags

Tag name	Arguments	Attributes	Description
@author	comments	none	specifies the author of the file
@copyright	comments	none	specifies the copyright information for the file
@examples	comments	none	specifies examples of usage
@hidden	none	none	if present, indicates the file is not to be shown in the documentation
@history	comments	none	lists the history for the file
@private	none	none	if present, indicates the file should not be shown in user-level documentation (set with the *USER* keyword to IDLdoc)
@property	property name, comments	none	describes a property of a class, i.e., a keyword to *getProperty*, *setProperty*, or *init*
@version	comments	none	specifies the version of the file

The following tags are available for comments immediately preceding or following a routine header.

Table 8 – IDLdoc format style routine tags

Tag name	Arguments	Attributes	Description
@abstract	none	none	if present, indicates the method is not implemented and present only to specify the interface to subclasses' implementations
@author	comments	none	specifies the author of the routine
@bugs	comments	none	specifies any issues found in the routine
@categories	list	none	specifies a comma-separated list of category names
@copyright	comments	none	specifies the copyright for the routine
@customer_id	comments	none	specifies a customer ID for the routine

... continued on next page

Table 8 – IDLdoc format style routine tags (... continued)

Tag name	**Arguments**	**Attributes**	**Description**
@description	comments	none	a tag for the standard comments for a routine; will be appended to standard comments if both are present
@examples	comments	none	specifies examples of using the routine
@field	fieldname and comments	none	specifies the name of the field followed by a description of the field
@file_comments	comments	none	equivalent to the main section in file-level comments
@hidden	none	none	if present, indicate the routine should not be shown in the documentation
@hidden_file	none	none	if present, indicates the file containing this routine should not be shown in the documentation
@history	comments	none	specifies the history of the routine
@inherits	none	none	not used
@keyword	keyword name	see below	documents a keyword of the routine
@obsolete	none	none	if present, indicates the routine is obsolete
@param	param name	see below	documents a positional parameter of the routine
@post	comments	none	specifies any post-conditions of the routine
@pre	comments	none	specifies any pre-conditions of the routine
@private	none	none	if present, indicates the routine should not be shown in user-level documentation (set with the *USER* keyword to IDLdoc)
@private_file	comments	none	if present, indicates the file containing this routine should not shown in user-level documentation (set with the *USER* keyword to IDLdoc)
@requires	comments	none	specifies the IDL version of the routine; IDLdoc finds the routines requiring the highest IDL version and reports them on the warnings page
@returns	comments	none	specifies the return value of the function
@todo	comments	none	specifies any todo items left for the routine

... continued on next page

Table 8 – IDLdoc format style routine tags (... continued)

Tag name	Arguments	Attributes	Description
@uses	comments	none	specifies any other routines, classes, etc. needed by the routine
@Version	comments	none	specifies the version of the routine

The keyword and param tags above accept attributes. The available attributes are shown below.

Table 9 – IDLdoc format style tag attributes

Attribute name	Syntax	Description
in	`in`	indicates the parameter is an input
out	`out`	indicates the parameter is an output
optional	`optional`	indicates argument is optional
private	`private`	indicates argument is not shown if IDLdoc is run in user mode (*USER* keyword to IDLdoc is set)
hidden	`hidden`	indicates the argument is not to be shown
required	`required`	indicates argument is required
type	`type=comments`	IDL data type of the argument
default	`default=comments`	default value of the argument

The tags available in an overview file describe the entire library. There are a few tags shared with the file tags and the additional *@dir* tag which provides a simple table of contents for the directories in the library.

Table 10 – rst format style tags for overview files

Tag name	Arguments	Attributes	Description
@author	comments	none	specifies the author of the library
@copyright	comments	none	specifies the copyright for the library
@dir	dir	none	lists directory in the library along with a description for each
@history	comments	none	specifies the history of the library
@version	comments	none	specifies the version of the library

Directory overview files also have a few tags shared with file tags.

Table 11 – rst format style tags for overview files

Tag name	Arguments	Attributes	Description
@author	comments	none	specifies the author of the files in the directory
@copyright	comments	none	specifies the copyright for the files in the directory
@hidden	none	none	if present, indicate the routine should not be shown in the documentation
@history	comments	none	specifies the history of the library
@private	none	none	if present, indicates the directory should not be shown in user-level documentation (set with the *USER* keyword to IDLdoc)

IDL format style

The IDL format style attempts to extract information from code using the IDL template, i.e., the form shown in *template.pro* in the *examples* directory of the IDL distribution.

```
;+
; NAME:
;   ROUTINE_NAME
;
; PURPOSE:
;   Tell what your routine does here.  I like to start with the words:
;   "This function (or procedure) ..."
;   Try to use the active, present tense.
;
; CATEGORY:
;   Put a category (or categories) here.  For example:
;  Widgets.
;
; CALLING SEQUENCE:
;   Write the calling sequence here. Include only positional parameters
;   (i.e., NO KEYWORDS). For procedures, use the form:
;
; ROUTINE_NAME, Parameter1, Parameter2, Foobar
;
; Note that the routine name is ALL CAPS and arguments have Initial
; Caps.  For functions, use the form:
```

```
;
; Result = FUNCTION_NAME(Parameter1, Parameter2, Foobar)
;
; Always use the "Result = " part to begin. This makes it super-obvious
; to the user that this routine is a function!
;
; INPUTS:
;   Parm1:  Describe the positional input parameters here. Note again
;     that positional parameters are shown with Initial Caps.
;
; OPTIONAL INPUTS:
;   Parm2:  Describe optional inputs here. If you don't have any, just
;     delete this section.
;
; KEYWORD PARAMETERS:
;   KEY1:  Document keyword parameters like this. Note that the keyword
;     is shown in ALL CAPS!
;
;   KEY2:  Yet another keyword. Try to use the active, present tense
;     when describing your keywords.  For example, if this keyword
;     is just a set or unset flag, say something like:
;     "Set this keyword to use foobar subfloatation. The default
;     is foobar superfloatation."
;
; OUTPUTS:
;   Describe any outputs here.  For example, "This function returns the
;   foobar superflimpt version of the input array."  This is where you
;   should also document the return value for functions.
;
; OPTIONAL OUTPUTS:
;   Describe optional outputs here.  If the routine doesn't have any,
;   just delete this section.
;
; COMMON BLOCKS:
;   BLOCK1:  Describe any common blocks here. If there are no COMMON
;     blocks, just delete this entry.
;
; SIDE EFFECTS:
;   Describe "side effects" here.  There aren't any?  Well, just delete
;   this entry.
;
; RESTRICTIONS:
```

```
;   Describe any "restrictions" here.  Delete this section if there are
;   no important restrictions.
;
; PROCEDURE:
;   You can describe the foobar superfloatation method being used here.
;   You might not need this section for your routine.
;
; EXAMPLE:
;   Please provide a simple example here. An example from the
;   DIALOG_PICKFILE documentation is shown below. Please try to
;   include examples that do not rely on variables or data files
;   that are not defined in the example code. Your example should
;   execute properly if typed in at the IDL command line with no
;   other preparation.
;
;       Create a DIALOG_PICKFILE dialog that lets users select only
;       files with the extension `pro'. Use the `Select File to Read'
;       title and store the name of the selected file in the variable
;       file. Enter:
;
;       file = DIALOG_PICKFILE(/READ, FILTER = '*.pro')
;
; MODIFICATION HISTORY:
;    Written by:  Your name here, Date.
;    July, 1994  Any additional mods get described here.  Remember to
;      change the stuff above if you add a new keyword or
;      something!
;-
```

The routine and file headings are shown in the table below.

Table 12 – IDL format style routine and file headings

Heading name	**Description**
calling sequence	calling sequence for the routine; unneeded since IDLdoc gets the calling sequence from the routine declaration
category	list of comma or period separated categories
common blocks	List common blocks, as in: `BLOCK1: description.`

... continued on next page

Table 12 – IDL format style routine and file headings (... continued)

Heading name	**Description**
example	list a simple example

inputs	list positional input parameters here as:
	`Param1: describe param1 here`
	
	`Param2: describe param2 here`

keyword parameters	document the keyword parameters here, listed as:
	`KEY1: key1 description`
	
	`KEY2: key2 description`

modification history	list history of modifications to the routine:
	`Written by: author name`
	`July 1994 Describe modifications done on this`
	`          date`

name	name of the routine; unneeded since IDLdoc gets the name of the routine from the routine declaration
optional inputs	list optional input parameters here, like: `Param3: describe param3 here`
optional outputs	describe the optional outputs here
outputs	documentation of the return value
procedure	describe/cite any algorithms being used in this routine
purpose	main description of the routine
restrictions	describe restrictions
side effects	describe side effects

There are no special headers for overview files or directory overview files using the IDL format style.

Markup styles

Markup styles specify annotations of text comments. The valid markup styles are: “rst”, “verbatim”, and “preformattted”.

rst markup style

The *rst* markup style is the default markup style for the *rst* format style.

Table 13 – rst markup style

Feature	**Description**

paragraphs	Paragraphs are created by simply skipping a line:
	`; Merges a string array into a single string separated by`
	`; carriage return/linefeeds.`
	`;`
	`; Defaults to use just linefeed on UNIX platforms and both`
	`; carriage returns and linefeeds on Windows platforms`
	`; unless the UNIX or WINDOWS keywords are set to force a`
	`; particular separator.`

code	To place a block of code into the documentation, end a line with `::`, skip a line, indent the block of code, and skip another line:
	`; Set the decomposed mode, if available in the current`
	`; graphics device i.e. equivalent to::`
	`;`
	`;    device, get_decomposed=oldDec`
	`;    device, decomposed=dec`
	`;`
	`; The main advantage of this routine is that it can be used`
	`; with any graphics device; it will be ignored in devices`
	`; which don't support it.`

... continued on next page

Table 13 – rst markup style (... continued)

Feature	**Description**

links	Another common annotation is to place a link in the documentation. For example, to link "http://michaelgalloy.com" to the phrase "my website", simply do:
	`` ; Check out `my website <http://michaelgalloy.com>`. ``
	But often, links are to other items in the documentation. For example, the comments for a routine, might briefly mention some of its keywords and it would be convenient to link to the documentation for these keywords. In this case, just put the method names in backticks like:
	`; :Returns:`
	`` ;    Returns a triple as a `bytarr(3)` or `bytarr(3, n)` by ``
	`;    default if a single color name or n color names are`
	`;    given. Returns a decomposed color index as a long or`
	`` ;    lonarr(n) if `INDEX` keyword is set. ``
	`;`
	`` ;    Returns a string array for the names if `NAMES` ``
	`;    keyword is set.`
	IDL will search for a name matching the quoted string and link to the closest one it finds. If the name is not found, as in `bytarr(3)` above, it will simply be displayed in a monospace space font as code.

headings	Different level headers can be added to comments, particularly useful for *.idldoc* files. Just underline with `-`, `=`, or `~`. For example, the following beginning to an *.idldoc* file, creates a level 1 header "TxDAP API Introduction", with a level 2 header "Basic Use" immediately after:
	`TxDAP API Introduction`
	`======================`
	
	`Basic Use`
	`---------`
	The order of use of the underlining determines the level of the header: the first underlined header is assumed to be level 1. The second, unless it is the same as the first, is assumed to be level 2, etc. From then on, titles underlined with "=" are level 1 headers and those underlined with "-" are level 2 headers.

... continued on next page

Table 13 – rst markup style (... continued)

Feature	Description
images	The "image" directive allows images to be placed into comments. To use, put the following on the end of a line: `.. image:: filename` where *filename* is any image file format read by *READ_IMAGE*. The *filename* specified will be copied into the output directory.
embed objects	The "embed" directive allows *.svg* files to be embedded in the documentation. To use, put the following on the end of a line: `.. embed:: filename`
HTML include	HTML can be included directly in the output via the HTML directive: `.. html:: <a ref="http://example.com">example</a>`
page title	The "title" directive is available to provide a title for *.idldoc* files: `.. title:: cpt-city color tables` This title is used for the *.idldoc* file in the table of contents of available documentation.

verbatim markup style

The *verbatim* markup style is the default markup style for the *IDLdoc* or *IDL* format styles.

preformatted style

The *preformatted* markup style must be specified as a markup style, it is not the default for any format style. Comments are copied directly into the output and wrapped with markup to display them in a fixed width font.

Customizing output

The output produced by IDLdoc can be customized by modifying the template files provides in the *templates/* directory of the IDLdoc distribution.

Instead of modifying the existing templates, it is best to copy the templates and specify their location with the *TEMPLATE_LOCATION* keyword to *IDLDOC*. If you have multiple template families, the *TEMPLATE_PREFIX* keyword can be used to specify a string that prefixes each filename of the template family. For example, IDLdoc itself uses the "latex-" prefix to specify the templates used to produce LaTeX output.

If IDLdoc is intended to produce some type of output besides HTML, the *COMMENT_STYLE* keyword must be used to specify the engine to produce that type of output. IDLdoc provides the "html", "latex", and "rst" comment styles. Creating new comment style engines is beyond the scope of this reference guide.