NAME

Lintian::Output – Lintian messaging handling

SYNOPSIS

```
    # non-OO
    use Lintian::Output qw(:messages);

    $Lintian::Output::GLOBAL->verbosity_level(1);

    msg("Something interesting");
    v_msg("Something less interesting");
    debug_msg(3, "Something very specific");

    # OO
    use Lintian::Output;

    my $out = Lintian::Output->new;

    $out->verbosity_level(-1);
    $out->msg("Something interesting");
    $out->v_msg("Something less interesting");
    $out->debug_msg(3, "Something very specific");
```

DESCRIPTION

Lintian::Output is used for all interaction between lintian and the user. It is designed to be easily extensible via subclassing.

To simplify usage in the most common cases, many Lintian::Output methods can be used as class methods and will therefore automatically use the object `$Lintian::Output::GLOBAL` unless their first argument `isa('Lintian::Output')`.

ATTRIBUTES

The following fields impact the behavior of Lintian::Output.

color

Can take the values ''never'', ''always'', ''auto'' or ''html''.

Whether to colorize tags based on their severity. The default is ''never'', which never uses color. ''always'' will always use color, ''auto'' will use color only if the output is going to a terminal.

''html'' will output HTML <span> tags with a color style attribute (instead of ANSI color escape sequences).

colors
debug

If set to a positive integer, will enable all debug messages issued with a level lower or equal to its value.

issuedtags

Hash containing the names of tags which have been issued.

perf_debug
perf_log_fd
proc_id2tag_count
stdout

I/O handle to use for output of messages and tags. Defaults to `\*STDOUT`.

stderr

I/O handle to use for warnings. Defaults to `\*STDERR`.

showdescription
: Whether to show the description of a tag when printing it.

tty_hyperlinks

tag_display_limit
: Get/Set the number of times a tag is emitted per processable.

verbosity_level
: Determine how verbose the output should be. "0" is the default value (tags and msg only), "−1" is quiet (tags only) and "1" is verbose (tags, msg and v_msg).

CLASS/INSTANCE METHODS

These methods can be used both with and without an object. If no object is given, they will fall back to the `$Lintian::Output::GLOBAL` object.

`msg(@args)`
: Will output the strings given in `@args`, one per line, each line prefixed with 'N: '. Will do nothing if verbosity_level is less than 0.

`v_msg(@args)`
: Will output the strings given in `@args`, one per line, each line prefixed with 'N: '. Will do nothing unless verbosity_level is greater than 0.

`debug_msg($level, @args)`
: `$level` should be a positive integer.

  Will output the strings given in `@args`, one per line, each line prefixed with 'N: '. Will do nothing unless debug is set to a positive integer >= `$level`.

`warning(@args)`
: Will output the strings given in `@args` on stderr, one per line, each line prefixed with 'warning: '.

`perf_log(@args)`
: Like "v_msg", except output is possibly sent to a dedicated log file.

  Will output the strings given in `@args`, one per line. The lines will not be prefixed. Will do nothing unless perf_debug is set to a positive integer.

`delimiter()`
: Gives back a string that is usable for separating messages in the output. Note: This does not print anything, it just gives back the string, use with one of the methods above, e.g.

```
 v_msg('foo', delimiter(), 'bar');
```

`issued_tag($tag_name)`
: Indicate that the named tag has been issued. Returns a boolean value indicating whether the tag had previously been issued by the object.

`string($lead, @args)`
: TODO: Is this part of the public interface?

INSTANCE METHODS FOR SUBCLASSING

The following methods are only intended for subclassing and are only available as instance methods. The methods mentioned in "CLASS/INSTANCE METHODS" usually only check whether they should do anything at all (according to the values of verbosity_level and debug) and then call one of the following methods to do the actual printing. Almost all of them finally call **_print()** to do that. This convoluted scheme is necessary to be able to use the methods above as class methods and still make the behaviour overridable in subclasses.

`_message(@args)`
: Called by **msg()**, **v_msg()**, and **debug_msg()** to print the message.

`_warning(@args)`
Called by **warning()** to print the warning.

`_print($stream, $lead, @args)`
Called by **_message()**, **_warning()**, and **print_tag()** to do the actual printing.

If you override these three methods, you can change the calling convention for this method to pretty much whatever you want.

The version in Lintian::Output prints the strings in `@args`, one per line, each line preceded by `$lead` to the I/O handle given in `$stream`.

`_delimiter()`
Called by **delimiter()**.

`_do_color()`
Called by **print_tag()** to determine whether to produce colored output.

EXPORTS

Lintian::Output exports nothing by default, but the following export tags are available:

:messages
Exports all the methods in "CLASS/INSTANCE METHODS"

:util
Exports all the methods in "CLASS METHODS"

SEE ALSO

lintian (1)