

**ferror**

1.2.4

Generated by Doxygen 1.8.11

## Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
1.1	Introduction . . . . .	1
<b>2</b>	<b>Modules Index</b>	<b>2</b>
2.1	Modules List . . . . .	2
<b>3</b>	<b>Data Type Index</b>	<b>2</b>
3.1	Data Types List . . . . .	2
<b>4</b>	<b>Module Documentation</b>	<b>2</b>
4.1	ferror Module Reference . . . . .	2
4.1.1	Detailed Description . . . . .	3
4.1.2	Function/Subroutine Documentation . . . . .	3
4.2	ferror_c_binding Module Reference . . . . .	7
4.2.1	Detailed Description . . . . .	8
4.2.2	Function/Subroutine Documentation . . . . .	8
<b>5</b>	<b>Data Type Documentation</b>	<b>13</b>
5.1	ferror_c_binding::errorhandler Type Reference . . . . .	13
5.1.1	Detailed Description . . . . .	13
5.2	ferror::errors Type Reference . . . . .	13
5.2.1	Detailed Description . . . . .	15
	<b>Index</b>	<b>17</b>

## 1 Main Page

### 1.1 Introduction

FERROR is a library to assist with error handling in Fortran projects.

#### Author

Jason Christopherson

#### Version

1.2.4

## 2 Modules Index

### 2.1 Modules List

Here is a list of all documented modules with brief descriptions:

<a href="#">ferror</a>	
<b>ferror</b>	<a href="#">2</a>
<a href="#">ferror_c_binding</a>	
<b>ferror</b>	<a href="#">7</a>

## 3 Data Type Index

### 3.1 Data Types List

Here are the data types with brief descriptions:

<a href="#">ferror_c_binding::errorhandler</a>	
<b>A C compatible type encapsulating an errors object</b>	<a href="#">13</a>
<a href="#">ferror::errors</a>	
<b>Defines a type for managing errors and warnings</b>	<a href="#">13</a>

## 4 Module Documentation

### 4.1 ferror Module Reference

**ferror**

Data Types

- type [errors](#)  
*Defines a type for managing errors and warnings.*

## Functions/Subroutines

- pure character(len=:) function, allocatable `er_get_log_filename` (this)  
*Gets the name of the error log file.*
- subroutine `er_set_log_filename` (this, str)  
*Sets the name of the error log file.*
- subroutine `er_report_error` (this, fcn, msg, flag)  
*Reports an error condition to the user.*
- subroutine `er_report_warning` (this, fcn, msg, flag)  
*Reports a warning message to the user.*
- subroutine `er_log_error` (this, fcn, msg, flag)  
*Writes an error log file.*
- pure logical function `er_has_error_occurred` (this)  
*Tests to see if an error has been encountered.*
- subroutine `er_reset_error_status` (this)  
*Resets the error status flag to false, and the current error flag to zero.*
- pure logical function `er_has_warning_occurred` (this)  
*Tests to see if a warning has been encountered.*
- subroutine `er_reset_warning_status` (this)  
*Resets the warning status flag to false, and the current warning flag to zero.*
- pure integer function `er_get_error_flag` (this)  
*Gets the current error flag.*
- pure integer function `er_get_warning_flag` (this)  
*Gets the current warning flag.*
- pure logical function `er_get_exit_on_error` (this)  
*Gets a logical value determining if the application should be terminated when an error is encountered.*
- subroutine `er_set_exit_on_error` (this, x)  
*Sets a logical value determining if the application should be terminated when an error is encountered.*
- pure logical function `er_get_suppress_printing` (this)  
*Gets a logical value determining if printing of error and warning messages should be suppressed.*
- subroutine `er_set_suppress_printing` (this, x)  
*Sets a logical value determining if printing of error and warning messages should be suppressed.*

## 4.1.1 Detailed Description

**error**

## Purpose

Provides a series of error codes and error handling mechanisms.

## 4.1.2 Function/Subroutine Documentation

4.1.2.1 pure integer function `error::er_get_error_flag` ( class(errors), intent(in) *this* ) [private]

Gets the current error flag.

## Parameters

in	<i>this</i>	The errors object.
----	-------------	--------------------

**Returns**

The current error flag.

**4.1.2.2 pure logical function `error::er_get_exit_on_error ( class(errors), intent(in) this )` [private]**

Gets a logical value determining if the application should be terminated when an error is encountered.

**Parameters**

in	<i>this</i>	The errors object.
----	-------------	--------------------

**Returns**

Returns true if the application should be terminated; else, false.

**4.1.2.3 pure character(len = :) function, allocatable `error::er_get_log_filename ( class(errors), intent(in) this )`**

Gets the name of the error log file.

**Parameters**

in	<i>this</i>	The errors object.
----	-------------	--------------------

**Returns**

The filename.

**4.1.2.4 pure logical function `error::er_get_suppress_printing ( class(errors), intent(in) this )` [private]**

Gets a logical value determining if printing of error and warning messages should be suppressed.

**Parameters**

in	<i>this</i>	The errors object.
----	-------------	--------------------

**Returns**

True if message printing should be suppressed; else, false to allow printing.

**4.1.2.5 pure integer function `error::er_get_warning_flag ( class(errors), intent(in) this )` [private]**

Gets the current warning flag.

**Parameters**

in	<i>this</i>	The errors object.
----	-------------	--------------------

**Returns**

The current warning flag.

4.1.2.6 pure logical function `error::er_has_error_occurred ( class(errors), intent(in) this )` [private]

Tests to see if an error has been encountered.

**Parameters**

in	<i>this</i>	The errors object.
----	-------------	--------------------

**Returns**

Returns true if an error has been encountered; else, false.

4.1.2.7 pure logical function `error::er_has_warning_occurred ( class(errors), intent(in) this )` [private]

Tests to see if a warning has been encountered.

**Parameters**

in	<i>this</i>	The errors object.
----	-------------	--------------------

**Returns**

Returns true if a warning has been encountered; else, false.

4.1.2.8 subroutine `error::er_log_error ( class(errors), intent(in) this, character(len = *), intent(in) fcn, character(len = *), intent(in) msg, integer, intent(in) flag )` [private]

Writes an error log file.

**Parameters**

in	<i>this</i>	The errors object.
in	<i>fcn</i>	The name of the function or subroutine in which the error was encountered.
in	<i>msg</i>	The error message.
in	<i>flag</i>	The error flag.

4.1.2.9 subroutine `error::er_report_error ( class(errors), intent(inout) this, character(len = *), intent(in) fcn, character(len = *), intent(in) msg, integer, intent(in) flag )` [private]

Reports an error condition to the user.

**Parameters**

in, out	<i>this</i>	The errors object.
in	<i>fcn</i>	The name of the function or subroutine in which the error was encountered.

**Parameters**

in	<i>msg</i>	The error message.
in	<i>flag</i>	The error flag.

**Remarks**

The default behavior prints an error message, appends the supplied information to a log file, and terminates the program.

**4.1.2.10** `subroutine ferror::er_report_warning ( class(errors), intent(inout) this, character(len = *), intent(in) fcn, character(len = *), intent(in) msg, integer, intent(in) flag ) [private]`

Reports a warning message to the user.

**Parameters**

in, out	<i>this</i>	The errors object.
in	<i>fcn</i>	The name of the function or subroutine from which the warning was issued.
in	<i>msg</i>	The warning message.
in	<i>flag</i>	The warning flag.

**Remarks**

The default behavior prints the warning message, and returns control back to the calling code.

**4.1.2.11** `subroutine ferror::er_reset_error_status ( class(errors), intent(inout) this ) [private]`

Resets the error status flag to false, and the current error flag to zero.

**Parameters**

in, out	<i>this</i>	The errors object.
---------	-------------	--------------------

**4.1.2.12** `subroutine ferror::er_reset_warning_status ( class(errors), intent(inout) this ) [private]`

Resets the warning status flag to false, and the current warning flag to zero.

**Parameters**

in, out	<i>this</i>	The errors object.
---------	-------------	--------------------

**4.1.2.13** `subroutine ferror::er_set_exit_on_error ( class(errors), intent(inout) this, logical, intent(in) x ) [private]`

Sets a logical value determining if the application should be terminated when an error is encountered.

## Parameters

<code>in, out</code>	<code>this</code>	The errors object.
<code>in</code>	<code>x</code>	Set to true if the application should be terminated when an error is reported; else, false.

4.1.2.14 `subroutine ferror::er_set_log_filename ( class(errors), intent(inout) this, character(len = :), allocatable str )`  
`[private]`

Sets the name of the error log file.

## Parameters

<code>in, out</code>	<code>this</code>	The errors object.
<code>in</code>	<code>str</code>	The filename.

4.1.2.15 `subroutine ferror::er_set_suppress_printing ( class(errors), intent(inout) this, logical, intent(in) x )` `[private]`

Sets a logical value determining if printing of error and warning messages should be suppressed.

## Parameters

<code>in, out</code>	<code>this</code>	The errors object.
<code>in</code>	<code>x</code>	Set to true if message printing should be suppressed; else, false to allow printing.

4.2 `ferror_c_binding` Module Reference**ferror**

## Data Types

- type `errorhandler`  
*A C compatible type encapsulating an errors object.*

## Functions/Subroutines

- subroutine `alloc_errorhandler` (obj)  
*Initializes a new error handler object.*
- subroutine `free_errorhandler` (obj)  
*Frees resources held by the errorhandler object.*
- subroutine `get_errorhandler` (obj, eobj)  
*Retrieves the errors object from the C compatible data structure.*
- subroutine `get_log_filename` (err, fname, nfname)  
*Gets the name of the error log file.*
- subroutine `set_log_filename` (err, fname)  
*Sets the error log filename.*
- subroutine `report_error` (err, fcn, msg, flag)



- Reports an error condition to the user.*
- subroutine `report_warning` (err, fcn, msg, flag)
- Reports a warning condition to the user.*
- subroutine `log_error` (err, fcn, msg, flag)
- Writes an error log file.*
- logical(c\_bool) function `has_error_occurred` (err)
- Tests to see if an error has been encountered.*
- subroutine `reset_error_status` (err)
- Resets the error status flag to false, and the current error flag to zero.*
- logical(c\_bool) function `has_warning_occurred` (err)
- Tests to see if a warning has been encountered.*
- subroutine `reset_warning_status` (err)
- Resets the warning status flag to false, and the current warning flag to zero.*
- integer(c\_int) function `get_error_flag` (err)
- Gets the current error flag.*
- integer(c\_int) function `get_warning_flag` (err)
- Gets the current warning flag.*
- logical(c\_bool) function `get_exit_on_error` (err)
- Gets a logical value determining if the application should be terminated when an error is encountered.*
- subroutine `set_exit_on_error` (err, x)
- Sets a logical value determining if the application should be terminated when an error is encountered.*
- logical(c\_bool) function `get_suppress_printing` (err)
- Gets a logical value determining if printing of error and warning messages should be suppressed.*
- subroutine `set_suppress_printing` (err, x)
- Sets a logical value determining if printing of error and warning messages should be suppressed.*
- character(len=:) function, allocatable `cstr_2_fstr` (cstr)
- Copies a C string (null terminated) to a Fortran string.*
- subroutine `fstr_2_cstr` (fstr, cstr, csize)
- Copies a Fortran string into a C string.*

#### 4.2.1 Detailed Description

##### **ferror**

##### **Purpose**

Provides C bindings to the ferror library.

#### 4.2.2 Function/Subroutine Documentation

##### 4.2.2.1 subroutine `ferror_c_binding::alloc_errorhandler` ( type(errorhandler), intent(inout) *obj* )

Initializes a new error handler object.

##### **Parameters**

<code>in</code>	<code>obj</code>	The errorhandler object to allocate.
-----------------	------------------	--------------------------------------

4.2.2.2 `character(len = :)` function, allocatable `error_c_binding::cstr_2_fstr` ( `character(kind = c_char)`, `dimension(*)`, `intent(in) cstr` )

Copies a C string (null terminated) to a Fortran string.

#### Parameters

<code>in</code>	<code>cstr</code>	The null-terminated C string.
-----------------	-------------------	-------------------------------

#### Returns

The Fortran copy.

4.2.2.3 subroutine `error_c_binding::free_errorhandler` ( `type(errorhandler)`, `intent(inout)`, `target obj` )

Frees resources held by the errorhandler object.

#### Parameters

<code>in, out</code>	<code>obj</code>	The errorhandler object.
----------------------	------------------	--------------------------

4.2.2.4 subroutine `error_c_binding::fstr_2_cstr` ( `character(len = *)`, `intent(in) fstr`, `character(kind = c_char)`, `dimension(*)`, `intent(out) cstr`, `integer`, `intent(inout) csize` )

Copies a Fortran string into a C string.

#### Parameters

<code>in</code>	<code>fstr</code>	The Fortran string to copy.
<code>out</code>	<code>cstr</code>	The null-terminated C string.
<code>in, out</code>	<code>csz</code>	On input, the size of the character buffer <code>cstr</code> . On output, the actual number of characters (not including the null character) written to <code>cstr</code> .

4.2.2.5 `integer(c_int)` function `error_c_binding::get_error_flag` ( `type(errorhandler)`, `intent(in) err` )

Gets the current error flag.

#### Parameters

<code>in</code>	<code>err</code>	The errorhandler object.
-----------------	------------------	--------------------------

#### Returns

The current error flag.

4.2.2.6 subroutine `error_c_binding::get_errorhandler` ( `type(errorhandler)`, `intent(in)`, `target obj`, `type(errors)`, `intent(out)`, `pointer eobj` )

Retrieves the errors object from the C compatible data structure.

**Parameters**

in	<i>obj</i>	The C compatible errorhandler data structure.
out	<i>eobj</i>	The resulting errors object.

**4.2.2.7 logical(c\_bool) function `error_c_binding::get_exit_on_error ( type(errorhandler), intent(in) err )`**

Gets a logical value determining if the application should be terminated when an error is encountered.

**Parameters**

in	<i>err</i>	The errorhandler object.
----	------------	--------------------------

**Returns**

Returns true if the application should be terminated; else, false.

**4.2.2.8 subroutine `error_c_binding::get_log_filename ( type(errorhandler), intent(in) err, character(kind = c_char), dimension(*), intent(out) fname, integer(c_int), intent(inout) nfname )`**

Gets the name of the error log file.

**Parameters**

in	<i>err</i>	The errorhandler object.
out	<i>fname</i>	A character buffer where the filename will be written. It is recommended that this be in the neighborhood of 256 elements.
in, out	<i>nfname</i>	On input, the actual size of the buffer. Be sure to leave room for the null terminator character. On output, the actual numbers of characters written to <i>fname</i> (not including the null character).

**4.2.2.9 logical(c\_bool) function `error_c_binding::get_suppress_printing ( type(errorhandler), intent(in) err )`**

Gets a logical value determining if printing of error and warning messages should be suppressed.

**Parameters**

in	<i>err</i>	The errorhandler object.
----	------------	--------------------------

**Returns**

True if message printing should be suppressed; else, false to allow printing.

**4.2.2.10 integer(c\_int) function `error_c_binding::get_warning_flag ( type(errorhandler), intent(in) err )`**

Gets the current warning flag.

**Parameters**

in	<i>err</i>	The errorhandler object.
----	------------	--------------------------

**Returns**

The current warning flag.

**4.2.2.11** `logical(c_bool) function error_c_binding::has_error_occurred ( type(errorhandler), intent(in) err )`

Tests to see if an error has been encountered.

**Parameters**

in	<i>err</i>	A pointer to the error handler object.
----	------------	--

**Returns**

Returns true if an error has been encountered; else, false.

**4.2.2.12** `logical(c_bool) function error_c_binding::has_warning_occurred ( type(errorhandler), intent(in) err )`

Tests to see if a warning has been encountered.

**Parameters**

in	<i>err</i>	The errorhandler object.
----	------------	--------------------------

**Returns**

Returns true if a warning has been encountered; else, false.

**4.2.2.13** `subroutine error_c_binding::log_error ( type(errorhandler), intent(in) err, character(kind = c_char), intent(in) fcn, character(kind = c_char), intent(in) msg, integer(c_int), intent(in), value flag )`

Writes an error log file.

**Parameters**

in	<i>err</i>	The errorhandler object.
in	<i>fcn</i>	The name of the function or subroutine in which the error was encountered.
in	<i>msg</i>	The error message.
in	<i>flag</i>	The error flag.

**4.2.2.14** `subroutine error_c_binding::report_error ( type(errorhandler), intent(inout) err, character(kind = c_char), intent(in) fcn, character(kind = c_char), intent(in) msg, integer(c_int), intent(in), value flag )`

Reports an error condition to the user.

**Parameters**

<i>in, out</i>	<i>err</i>	A pointer to the error handler object.
<i>in</i>	<i>fcn</i>	The name of the function or subroutine in which the error was encountered.
<i>in</i>	<i>msg</i>	The error message.
<i>in</i>	<i>flag</i>	The error flag.

**4.2.2.15** `subroutine ferror_c_binding::report_warning ( type(errorhandler), intent(inout) err, character(kind = c_char), intent(in) fcn, character(kind = c_char), intent(in) msg, integer(c_int), intent(in), value flag )`

Reports a warning condition to the user.

**Parameters**

<i>in, out</i>	<i>err</i>	The errorhandler object.
<i>in</i>	<i>fcn</i>	The name of the function or subroutine in which the warning was encountered.
<i>in</i>	<i>msg</i>	The warning message.
<i>in</i>	<i>flag</i>	The warning flag.

**4.2.2.16** `subroutine ferror_c_binding::reset_error_status ( type(errorhandler), intent(inout) err )`

Resets the error status flag to false, and the current error flag to zero.

**Parameters**

<i>in, out</i>	<i>err</i>	The errorhandler object.
----------------	------------	--------------------------

**4.2.2.17** `subroutine ferror_c_binding::reset_warning_status ( type(errorhandler), intent(inout) err )`

Resets the warning status flag to false, and the current warning flag to zero.

**Parameters**

<i>in, out</i>	<i>err</i>	The errorhandler object.
----------------	------------	--------------------------

**4.2.2.18** `subroutine ferror_c_binding::set_exit_on_error ( type(errorhandler), intent(inout) err, logical(c_bool), intent(in), value x )`

Sets a logical value determining if the application should be terminated when an error is encountered.

**Parameters**

<i>in, out</i>	<i>err</i>	The errorhandler object.
<i>in</i>	<i>x</i>	Set to true if the application should be terminated when an error is reported; else, false.

**4.2.2.19** `subroutine ferror_c_binding::set_log_filename ( type(errorhandler), intent(inout) err, character(kind = c_char), dimension(*), intent(in) fname )`

Sets the error log filename.

#### Parameters

<i>in, out</i>	<i>err</i>	The errorhandler object.
<i>in</i>	<i>fname</i>	A null-terminated string containing the filename.

**4.2.2.20** `subroutine ferror_c_binding::set_suppress_printing ( type(errorhandler), intent(inout) err, logical(c_bool), intent(in), value x )`

Sets a logical value determining if printing of error and warning messages should be suppressed.

#### Parameters

<i>in, out</i>	<i>err</i>	The errorhandler object.
<i>in</i>	<i>x</i>	Set to true if message printing should be suppressed; else, false to allow printing.

## 5 Data Type Documentation

### 5.1 `ferror_c_binding::errorhandler` Type Reference

A C compatible type encapsulating an errors object.

#### Public Attributes

- `type(c_ptr)` *ptr*  
*A pointer to the errors object.*
- `integer(c_int)` *n*  
*The size of the errors object, in bytes.*

#### 5.1.1 Detailed Description

A C compatible type encapsulating an errors object.

The documentation for this type was generated from the following file:

- `/home/jason/Documents/Code/ferror/src/ferror_c_binding.f90`

### 5.2 `ferror::errors` Type Reference

Defines a type for managing errors and warnings.

## Public Member Functions

- procedure, public `get_log_filename => er_get_log_filename`  
*Gets the name of the error log file.*
- procedure, public `set_log_filename => er_set_log_filename`  
*Sets the name of the error log file.*
- procedure, public `report_error => er_report_error`  
*Reports an error condition to the user.*
- procedure, public `report_warning => er_report_warning`  
*Reports a warning message to the user.*
- procedure, public `log_error => er_log_error`  
*Writes an error log file.*
- procedure, public `has_error_occurred => er_has_error_occurred`  
*Tests to see if an error has been encountered.*
- procedure, public `reset_error_status => er_reset_error_status`  
*Resets the error status flag to false.*
- procedure, public `has_warning_occurred => er_has_warning_occurred`  
*Tests to see if a warning has been encountered.*
- procedure, public `reset_warning_status => er_reset_warning_status`  
*Resets the warning status flag to false.*
- procedure, public `get_error_flag => er_get_error_flag`  
*Gets the current error flag.*
- procedure, public `get_warning_flag => er_get_warning_flag`  
*Gets the current warning flag.*
- procedure, public `get_exit_on_error => er_get_exit_on_error`  
*Gets a logical value determining if the application should be terminated when an error is encountered.*
- procedure, public `set_exit_on_error => er_set_exit_on_error`  
*Sets a logical value determining if the application should be terminated when an error is encountered.*
- procedure, public `get_suppress_printing => er_get_suppress_printing`  
*Gets a logical value determining if printing of error and warning messages should be suppressed.*
- procedure, public `set_suppress_printing => er_set_suppress_printing`  
*Sets a logical value determining if printing of error and warning messages should be suppressed.*

## Public Attributes

- character(len=256) `m_fname` = "error\_log.txt"  
*A maximum of 256 character error log filename.*
- logical `m_founderror` = .false.  
*Found an error.*
- logical `m_foundwarning` = .false.  
*Found a warning.*
- integer `m_errorflag` = 0  
*The error flag.*
- integer `m_warningflag` = 0  
*The warning flag.*
- logical `m_exitonerror` = .true.  
*Terminate the application on error.*
- logical `m_suppressprinting` = .false.  
*Suppress printing of error and warning messages.*

### 5.2.1 Detailed Description

Defines a type for managing errors and warnings.

The documentation for this type was generated from the following file:

- `/home/jason/Documents/Code/ferror/src/ferror.f90`





## Index

- alloc\_errorhandler
  - [ferror\\_c\\_binding](#), [8](#)
- cstr\_2\_fstr
  - [ferror\\_c\\_binding](#), [8](#)
- er\_get\_error\_flag
  - [ferror](#), [3](#)
- er\_get\_exit\_on\_error
  - [ferror](#), [4](#)
- er\_get\_log\_filename
  - [ferror](#), [4](#)
- er\_get\_suppress\_printing
  - [ferror](#), [4](#)
- er\_get\_warning\_flag
  - [ferror](#), [4](#)
- er\_has\_error\_occurred
  - [ferror](#), [5](#)
- er\_has\_warning\_occurred
  - [ferror](#), [5](#)
- er\_log\_error
  - [ferror](#), [5](#)
- er\_report\_error
  - [ferror](#), [5](#)
- er\_report\_warning
  - [ferror](#), [6](#)
- er\_reset\_error\_status
  - [ferror](#), [6](#)
- er\_reset\_warning\_status
  - [ferror](#), [6](#)
- er\_set\_exit\_on\_error
  - [ferror](#), [6](#)
- er\_set\_log\_filename
  - [ferror](#), [7](#)
- er\_set\_suppress\_printing
  - [ferror](#), [7](#)
- [ferror](#), [2](#)
  - [er\\_get\\_error\\_flag](#), [3](#)
  - [er\\_get\\_exit\\_on\\_error](#), [4](#)
  - [er\\_get\\_log\\_filename](#), [4](#)
  - [er\\_get\\_suppress\\_printing](#), [4](#)
  - [er\\_get\\_warning\\_flag](#), [4](#)
  - [er\\_has\\_error\\_occurred](#), [5](#)
  - [er\\_has\\_warning\\_occurred](#), [5](#)
  - [er\\_log\\_error](#), [5](#)
  - [er\\_report\\_error](#), [5](#)
  - [er\\_report\\_warning](#), [6](#)
  - [er\\_reset\\_error\\_status](#), [6](#)
  - [er\\_reset\\_warning\\_status](#), [6](#)
  - [er\\_set\\_exit\\_on\\_error](#), [6](#)
  - [er\\_set\\_log\\_filename](#), [7](#)
  - [er\\_set\\_suppress\\_printing](#), [7](#)
- [ferror::errors](#), [13](#)
- [ferror\\_c\\_binding](#), [7](#)
  - [alloc\\_errorhandler](#), [8](#)
  - [cstr\\_2\\_fstr](#), [8](#)
  - [free\\_errorhandler](#), [9](#)
  - [fstr\\_2\\_cstr](#), [9](#)
  - [get\\_error\\_flag](#), [9](#)
  - [get\\_errorhandler](#), [9](#)
  - [get\\_exit\\_on\\_error](#), [10](#)
  - [get\\_log\\_filename](#), [10](#)
  - [get\\_suppress\\_printing](#), [10](#)
  - [get\\_warning\\_flag](#), [10](#)
  - [has\\_error\\_occurred](#), [11](#)
  - [has\\_warning\\_occurred](#), [11](#)
  - [log\\_error](#), [11](#)
  - [report\\_error](#), [11](#)
  - [report\\_warning](#), [12](#)
  - [reset\\_error\\_status](#), [12](#)
  - [reset\\_warning\\_status](#), [12](#)
  - [set\\_exit\\_on\\_error](#), [12](#)
  - [set\\_log\\_filename](#), [12](#)
  - [set\\_suppress\\_printing](#), [13](#)
  - [ferror\\_c\\_binding::errorhandler](#), [13](#)
  - [free\\_errorhandler](#)
    - [ferror\\_c\\_binding](#), [9](#)
  - [fstr\\_2\\_cstr](#)
    - [ferror\\_c\\_binding](#), [9](#)
  - [get\\_error\\_flag](#)
    - [ferror\\_c\\_binding](#), [9](#)
  - [get\\_errorhandler](#)
    - [ferror\\_c\\_binding](#), [9](#)
  - [get\\_exit\\_on\\_error](#)
    - [ferror\\_c\\_binding](#), [10](#)
  - [get\\_log\\_filename](#)
    - [ferror\\_c\\_binding](#), [10](#)
  - [get\\_suppress\\_printing](#)
    - [ferror\\_c\\_binding](#), [10](#)
  - [get\\_warning\\_flag](#)
    - [ferror\\_c\\_binding](#), [10](#)
  - [has\\_error\\_occurred](#)
    - [ferror\\_c\\_binding](#), [11](#)
  - [has\\_warning\\_occurred](#)
    - [ferror\\_c\\_binding](#), [11](#)
  - [log\\_error](#)
    - [ferror\\_c\\_binding](#), [11](#)
  - [report\\_error](#)
    - [ferror\\_c\\_binding](#), [11](#)
  - [report\\_warning](#)
    - [ferror\\_c\\_binding](#), [12](#)
  - [reset\\_error\\_status](#)
    - [ferror\\_c\\_binding](#), [12](#)
  - [reset\\_warning\\_status](#)
    - [ferror\\_c\\_binding](#), [12](#)
  - [set\\_exit\\_on\\_error](#)

[ferror\\_c\\_binding](#), [12](#)  
set\_log\_filename  
    [ferror\\_c\\_binding](#), [12](#)  
set\_suppress\_printing  
    [ferror\\_c\\_binding](#), [13](#)