

# **iLOG3**

## **Reference Manual**

Revised version

版本	日期	修订人	内容
1.0.0	2014-01-12	calvin	Create
1.0.1	2014-02-03	calvin	Add Log handle collection
1.0.2	2014-02-09	calvin	Add The configuration file interface layer
1.0.3	2014-02-14	calvin	Add SetFilterLogFunc SetBeforeRotateFileFunc SetAfterRotateFileFunc
1.0.4	2014-05-18	calvin	Add The log output MACRO LOG_OUTPUT_NOSET

## Indexes

1	Macros .....	6
1.1	Error code macros .....	6
1.2	The log output macros .....	6
1.3	Log level macros.....	6
1.4	Line logging combination style macros .....	7
1.5	Log options combination macros .....	8
1.6	Log rolling mode macros .....	8
2	Log handle functions.....	9
2.1	Admin log handle .....	9
2.1.1	Create log handle .....	9
2.1.2	Destruction of log handle.....	9
2.2	Handle the environment Settings .....	10
2.2.1	Set the log output.....	10
2.2.2	Set the current log filter level.....	10
2.2.3	Set line log style.....	11
2.3	Advanced handle environment Settings .....	11
2.3.1	Set the custom log filter callback function .....	11
2.3.2	Set the log handle options .....	11
2.3.3	Set the custom tag.....	12
2.3.4	Set log rolling mode .....	12
2.3.5	Set the log file rolling size.....	12
2.3.6	Set the size of the log file transfer file compression coefficient .....	13
2.3.7	Set the custom log file before the callback function.....	13
2.3.8	Set the custom log file after the callback function.....	13
2.3.9	Set line log buffer size .....	14
2.3.10	Set the hex piece of log buffer size.....	14
2.4	Line of log output .....	14
2.4.1	Write the line log with log level.....	14
2.4.2	Write line log with DEBUG level .....	15

2.4.3	Write line log with INFO level.....	15
2.4.4	Write line log with WARN level .....	16
2.4.5	Write line log with ERROR level.....	16
2.4.6	Write line log with FATAL level .....	16
2.5	Hex piece of log output .....	17
2.5.1	Write the hex piece log with log level .....	17
2.5.2	Write hex piece log with DEBUG level.....	17
2.5.3	Write hex piece log with INFO level .....	18
2.5.4	Write hex piece log with WARN level.....	18
2.5.5	Write hex piece log with ERROR level .....	19
2.5.6	Write hex piece log with FATAL level .....	19
3	The log handle collection functions .....	20
3.1	Admin log handle collection.....	20
3.1.1	Create log handle collection.....	20
3.1.2	Destruction of logging handle collection.....	20
3.2	Admin log log of a set of handle handle.....	21
3.2.1	Into a log to the log handle handle collection.....	21
3.2.2	From a log handle collection of pop up a log handle of the specified identification .....	21
3.2.3	From a query log handle the collection a log handle of the specified identification .....	22
3.2.4	Through a log in the collection of all log handle handle .....	22
4	Configuration file interface layer functions.....	23
4.1	Log handle.....	23
4.1.1	Handle to the build log from the config file .....	23
4.2	Log handle collection .....	23
4.2.1	The build log handle collection from the config file.....	23
5	Configuration of auxiliary function .....	24
5.1	Property transfer function.....	24
5.1.1	Log output type (string converted to an integer) .....	24

5.1.2	Log level types (string converted to an integer) .....	24
5.1.3	Log level types (integer is converted to a string) .....	24
5.1.4	Line log style (string converted to an integer) .....	25
5.1.5	Log option (string converted to an integer) .....	25
5.1.6	Log rolling mode (string is converted to an integer) .....	25
5.1.7	The log buffer (string converted to an integer) .....	26
6	Simple configuration file attributes list .....	26

# 1 Macros

## 1.1 Error code macros

MACRO	DESCRIPTION
LOG_RETURN_ERROR_ALLOC	Alloc memory failure
LOG_RETURN_ERROR_INTERNAL	Internal error. It's not your fault!
LOG_RETURN_ERROR_PARAMETER	Parameter error
LOG_RETURN_ERROR_NOTSUPPORT	Sorry, no support
LOG_RETURN_ERROR_CREATEFILE	Failed to create file
LOG_RETURN_ERROR_OPENFILE	Failed to open the file

## 1.2 The log output macros

MACRO	DESCRIPTION
LOG_OUTPUT_NOSET	Commonly used in subsequent change output filename
LOG_OUTPUT_STDOUT	The standard output
LOG_OUTPUT_STDERR	The standard error output
LOG_OUTPUT_SYSLOG	UNIX&Linux syslog 或 Windows WINDOWS EVENT
LOG_OUTPUT_FILE	File
LOG_OUTPUT_CALLBACK	Custom log output callback function

## 1.3 Log level macros

MACRO	DESCRIPTION
-------	-------------

LOG_LEVEL_DEBUG	Debug level
LOG_LEVEL_INFO	Common information level
LOG_LEVEL_WARN	Warning level
LOG_LEVEL_ERROR	Error level
LOG_LEVEL_FATAL	Serious error level  Often caused by a system problem can't continue to work error, the end of the process, lest destroy data
LOG_LEVEL_NOLOG	Don't need the output log

## 1.4 Line logging combination style macros

MACRO	DESCRIPTION
LOG_STYLE_DATE	DATE"YYYY-MM-DD"
LOG_STYLE_DATETIME	DATETIME"YYYY-MM-DD hh:mm:ss"
LOG_STYLE_DATETIMEMS	DATETIME"YYYY-MM-DD hh:mm:ss.6ms"  ( Date/time class macro mutexes, priority automatically selects the most information )
LOG_STYLE_LOGLEVEL	The log level
LOG_STYLE_PID	Process id
LOG_STYLE_TID	Thread id
LOG_STYLE_SOURCE	"Source file:line number"
LOG_STYLE_FORMAT	Log section
LOG_STYLE_NEWLINE	Newline
LOG_STYLE_CUSTLABEL1	Custom tag 1
LOG_STYLE_CUSTLABEL2	Custom tag 2
LOG_STYLE_CUSTLABEL3	Custom tag 3
LOG_STYLE_CALLBACK	( Using a custom log style callback function )
LOG_STYLE_...	( Waiting for you to expand... )

## 1.5 Log options combination macros

MACRO	DESCRIPTION
LOG_OPTION_OPEN_AND_CLOSE	Every time open the log, write the log, close the log
LOG_OPTION_CHANGE_TEST	Detect file changes
LOG_OPTION_OPEN_ONCE	Log open not to close
LOG_OPTION_SET_OUTPUT_BY_FILENAME	Cover the output type automatically according to the file name  If the file name is in front of the "# stdout#" prefix, prefix and cover the output type is LOG_OUTPUT_STDOUT  If the file name is in front of the "# stderr#" prefix, prefix and cover the output type is LOG_OUTPUT_STDERR  If the file name is in front of the "# syslog#" prefix, prefix and cover the output type is LOG_OUTPUT_SYSLOG
LOG_OPTION_FILENAME_APPEND_DOT_LOG	Log output filename automatically after plus ".log"

## 1.6 Log rolling mode macros

MACRO	DESCRIPTION
LOG_ROTATEMODE_NONE	Don't turn archives
LOG_ROTATEMODE_SIZE	按日志文件大小转档，和函数 SetLogRotateSize 配合使用 转档文件名格式"原日志文件名.序号"  According to the log file size to turn archives, and the function SetLogRotateSize cooperate to use



	Transfer file filename format " filename.serial-number"
LOG_ROTATEMODE_PER_DAY	According to the daily turn archives "filename.YYYYMMDD"
LOG_ROTATEMODE_PER_HOUR	According to the hours turn archives "filename.YYYYMMDD_HH"

## 2 Log handle functions

### 2.1 Admin log handle

#### 2.1.1 Create log handle

**PROTOTYPE:** LOG \*CreateLogHandle();

**PARAMETER:** (无)

**RETURN:** Successful, returns a handle to the newly created log  
Failure, returns NULL, general memory failure for application

**REMARK:**

#### 2.1.2 Destruction of log handle

**PROTOTYPE:** void DestroyLogHandle( LOG \*g );

**PARAMETER:** INPUT LOG \*g log handle

**RETURN:** (无)

**REMARK:**

## 2.2 Handle the environment Settings

### 2.2.1 Set the log output

**PROTOTYPE:** int SetLogOutput ( LOG \*g , int output , char \*log\_pathfilename , funcOpenLog  
\*pfuncOpenLogFirst , funcOpenLog \*pfuncOpenLog , funcWriteLog \*pfuncWriteLog ,  
funcChangeTest \*pfuncChangeTest , funcCloseLog \*pfuncCloseLog , funcCloseLog  
\*pfuncCloseLogFinally );

**PARAMETER:** INPUT LOG \*g log handle

INPUT int output see log output macro

INPUT char \*log\_pathfilename Log file name, allowing embedded environment  
variable by "\$...\$"

INPUT LOG\_OPTION\_OPEN\_ONCE or LOG\_OPTION\_CHANGE\_TEST mode, the  
initialization calls when pfuncOpenLogFirst, then pfuncWriteLog, call pfuncCloseLogFinally finally  
LOG\_OPTION\_OPEN\_AND\_CLOSE mode, the process of writing log call sequence pfuncOpenLog、  
pfuncWriteLog、 pfuncCloseLog

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.2.2 Set the current log filter level

**PROTOTYPE:** int SetLogLevel( LOG \*g , int log\_level );

**PARAMETER:** INPUT LOG \*g log handle

INPUT int log\_level see log level macro

**RETURN:** return 0

**REMARK:**

### 2.2.3 Set line log style

**PROTOTYPE:** int SetLogStyles( LOG \*g , long log\_styles , funcLogStyle \*pfuncLogStyle );

**PARAMETER:** **INPUT** LOG \*g log handle  
INPUT long log\_styles see log style macros  
INPUT funcLogStyle \*pfuncLogStyle

**RETURN:** return 0

**REMARK:**

## 2.3 Advanced handle environment Settings

### 2.3.1 Set the custom log filter callback function

**PROTOTYPE:** int SetFilterLogFunc( LOG \*g , funcFilterLog \*pfuncFilterLog );

**PARAMETER:** **INPUT** LOG \*g log handle  
INPUT funcFilterLog \*pfuncFilterLog Custom log filtered off function

**RETURN:** Successful, return 0  
Failure, returns non-zero, see error code macros

**REMARK:**

### 2.3.2 Set the log handle options

**PROTOTYPE:** int SetLogOptions( LOG \*g , int log\_options );

**PARAMETER:** **INPUT** LOG \*g log handle  
INPUT int log\_options see log options

**RETURN:** Successful, return 0  
Failure, returns non-zero, see error code macros

**REMARK:**

### 2.3.3 Set the custom tag

**PROTOTYPE:** int SetLogCustLabel( LOG \*g , int index , char \*cust\_label );

**PARAMETER:** INPUT LOG \*g log handle

INPUT int index Label index, from 1 to LOG\_MAXCNT\_CUST\_LABEL

INPUT char \*cust\_label Tag value, the longest LOG\_MAXLEN\_CUST\_LABEL

characters

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.3.4 Set log rolling mode

**PROTOTYPE:** int SetLogRotateMode( LOG \*g , int rotate\_mode );

**PARAMETER:** INPUT LOG \*g log handle

INOUT int rotate\_mode see log rolling mode macros

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.3.5 Set the log file rolling size

**PROTOTYPE:** int SetLogRotateSize( LOG \*g , long log\_rotate\_size );

**PARAMETER:** INPUT LOG \*g log handle

INPUT long log\_rotate\_size

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.3.6 Set the size of the log file transfer file compression coefficient

**PROTOTYPE:** int SetLogRotatePressureFactor( LOG \*g , long pressure\_factor );

**PARAMETER:** INPUT LOG \*g log handle

INPUT long pressure\_factor

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.3.7 Set the custom log file before the callback function

**PROTOTYPE:** int SetBeforeRotateFileFunc( LOG \*g , funcAfterRotateFile \*pfuncAfterRotateFile );

**PARAMETER:** INPUT LOG \*g log handle

INPUT funcAfterRotateFile \*pfuncAfterRotateFile Custom log file before the  
callback function

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.3.8 Set the custom log file after the callback function

**PROTOTYPE:** int SetAfterRotateFileFunc( LOG \*g , funcAfterRotateFile \*pfuncAfterRotateFile );

**PARAMETER:** INPUT LOG \*g log handle

INPUT funcAfterRotateFile \*pfuncAfterRotateFile Custom log file after the  
callback function

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:** Used to turn archives the post-processing, generally used to compress the log document

### 2.3.9 Set line log buffer size

**PROTOTYPE:** int SetLogBufferSize( LOG \*g , long log\_bufsize , long max\_log\_bufsize );

**PARAMETER:** **INPUT** LOG \*g log handle

INPUT long log\_bufsize

INPUT long max\_log\_bufsize

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.3.10 Set the hex piece of log buffer size

**PROTOTYPE:** int SetHexLogBufferSize( LOG \*g , long hexlog\_bufsize , long max\_log\_bufsize );

**PARAMETER:** **INPUT** LOG \*g log handle

INPUT long hexlog\_bufsize

INPUT long max\_log\_bufsize

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

## 2.4 Line of log output

### 2.4.1 Write the line log with log level

**PROTOTYPE:** int WriteLog( LOG \*g , char \*c\_filename , long c\_fileline , int log\_level , char

\*format , ... );

**PARAMETER:** INPUT LOG \*g log handle

INPUT char \*c\_filename The current source file name \_\_FILE\_\_

INPUT long c\_fileline The current source code line number \_\_LINE\_\_

INPUT int log\_level see log level macros

INPUT char \*format , ...

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

## 2.4.2 Write line log with DEBUG level

**PROTOTYPE:** int DebugLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*format , ... );

**PARAMETER:** INPUT LOG \*g log handle

INPUT char \*c\_filename The current source file name \_\_FILE\_\_

INPUT long c\_fileline The current source code line number \_\_LINE\_\_

INPUT char \*format , ...

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

## 2.4.3 Write line log with INFO level

**PROTOTYPE:** int InfoLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*format , ... );

**PARAMETER:** INPUT LOG \*g log handle

INPUT char \*c\_filename The current source file name \_\_FILE\_\_

INPUT long c\_fileline The current source code line number \_\_LINE\_\_

INPUT char \*format , ...

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

## 2.4.4 Write line log with WARN level

**PROTOTYPE:** int WarnLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*format , ... );

**PARAMETER:** **INPUT** LOG \*g log handle

INPUT char \*c\_filename The current source file name \_\_FILE\_\_

INPUT long c\_fileline The current source code line number \_\_LINE\_\_

INPUT char \*format , ...

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

## 2.4.5 Write line log with ERROR level

**PROTOTYPE:** int ErrorLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*format , ... );

**PARAMETER:** **INPUT** LOG \*g log handle

INPUT char \*c\_filename The current source file name \_\_FILE\_\_

INPUT long c\_fileline The current source code line number \_\_LINE\_\_

INPUT char \*format , ...

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

## 2.4.6 Write line log with FATAL level

**PROTOTYPE:** int FatalLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*format , ... );



**PARAMETER:** **INPUT** LOG \*g log handle

INPUT char \*c\_filename The current source file name \_\_FILE\_\_

INPUT long c\_fileline The current source code line number \_\_LINE\_\_

INPUT char \*format , ...

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

## 2.5 Hex piece of log output

### 2.5.1 Write the hex piece log with log level

**PROTOTYPE:** int WriteHexLog( LOG \*g , char \*c\_filename , long c\_fileline , int log\_level , char \*buffer , long buflen , char \*format , ... );

**PARAMETER:** **INPUT** LOG \*g log handle

INPUT char \*c\_filename The current source file name \_\_FILE\_\_

INPUT long c\_fileline The current source code line number \_\_LINE\_\_

INPUT int log\_level see log level macros

INPUT char \*buffer

INPUT long buflen

INPUT char \*format , ...

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.5.2 Write hex piece log with DEBUG level

**PROTOTYPE:** int DebugHexLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*buffer , long buflen , char \*format , ... );

**PARAMETER:** **INPUT** LOG \*g log handle

INPUT char \*c\_filename The current source file name \_\_FILE\_\_

INPUT long c\_fileline The current source code line number \_\_LINE\_\_

INPUT char \*buffer

INPUT long buflen

INPUT char \*format , ...

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.5.3 Write hex piece log with INFO level

**PROTOTYPE:** int InfoHexLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*buffer , long buflen , char \*format , ... );

**PARAMETER:** **INPUT** LOG \*g log handle

INPUT char \*c\_filename The current source file name \_\_FILE\_\_

INPUT long c\_fileline The current source code line number \_\_LINE\_\_

INPUT char \*buffer

INPUT long buflen

INPUT char \*format , ...

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 2.5.4 Write hex piece log with WARN level

**PROTOTYPE:** int WarnHexLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*buffer , long buflen , char \*format , ... );

**PARAMETER:** **INPUT** LOG \*g log handle

INPUT    char \*c\_filename    The current source file name \_\_FILE\_\_  
 INPUT    long c\_fileline        The current source code line number \_\_LINE\_\_  
 INPUT    char \*buffer  
 INPUT    long buflen  
 INPUT    char \*format , ...  
**RETURN:**    Successful, return 0  
              Failure, returns non-zero, see error code macros  
**REMARK:**

## 2.5.5 Write hex piece log with ERROR level

**PROTOTYPE:** int ErrorHexLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*buffer , long buflen , char \*format , ... );

**PARAMETER:** INPUT    LOG \*g    log handle  
                  INPUT    char \*c\_filename    The current source file name \_\_FILE\_\_  
                  INPUT    long c\_fileline        The current source code line number \_\_LINE\_\_  
                  INPUT    char \*buffer  
                  INPUT    long buflen  
                  INPUT    char \*format , ...  
**RETURN:**    Successful, return 0  
              Failure, returns non-zero, see error code macros  
**REMARK:**

## 2.5.6 Write hex piece log with FATAL level

**PROTOTYPE:** int FatalHexLog( LOG \*g , char \*c\_filename , long c\_fileline , char \*buffer , long buflen , char \*format , ... );

**PARAMETER:** INPUT    LOG \*g    log handle  
                  INPUT    char \*c\_filename    The current source file name \_\_FILE\_\_

INPUT    long c\_fileline        The current source code line number \_\_LINE\_\_

INPUT    char \*buffer

INPUT    long buflen

INPUT    char \*format , ...

**RETURN:**    Successful, return 0

             Failure, returns non-zero, see error code macros

**REMARK:**

## 3 The log handle collection functions

### 3.1 Admin log handle collection

#### 3.1.1 Create log handle collection

**PROTOTYPE:** LOGS \*CreateLogsHandle();

**PARAMETER:** (无)

**RETURN:**    Successful, returns a handle to the newly created log collection

             Failure, returns NULL, general memory failure for application

**REMARK:**

#### 3.1.2 Destruction of logging handle collection

**PROTOTYPE:** void DestroyLogsHandle( LOGS \*g );

**PARAMETER:** INPUT    LOGS \*gs log handle collection

**RETURN:**        (无)

**REMARK:**

## 3.2 Admin log log of a set of handle handle

### 3.2.1 Into a log to the log handle handle collection

**PROTOTYPE:** int AddLogToLogs( LOGS \*gs , char \*g\_id , LOG \*g );

**PARAMETER:** INPUT LOGS \*gs log handle collection

INPUT char \*g\_id

INPUT LOG \*g

**RETURN:** Successful, return 0

Failure, returns non-zero, see error code macros

**REMARK:**

### 3.2.2 From a log handle collection of pop up a log handle of the specified identification

**PROTOTYPE:** LOG \*RemoveOutLogFromLogs( LOGS \*gs , char \*g\_id );

**PARAMETER:** INPUT LOGS \*gs log handle collection

INPUT char \*g\_id

INPUT LOG \*g

**RETURN:** Successful, returns a handle to a pop-up log

Fails, the return NULL

**REMARK:** Pop-up log handle just remove the relationship with the logging handle collection, not destroyed, to be rid of the users themselves. Or destroy log handle set DestroyLogsHandle joint destruction of all the logs in handle.

### 3.2.3 From a query log handle the collection a log handle of the specified identification

**PROTOTYPE:** LOG \*GetLogFromLogs( LOGS \*gs , char \*g\_id );

**PARAMETER:** ( Same as above )

**RETURN:** ( Same as above )

**REMARK:**

### 3.2.4 Through a log in the collection of all log handle handle

**PROTOTYPE:** int TravelLogFromLogs( LOGS \*gs , long \*p\_index , char \*\*pp\_g\_id , LOG \*\*pp\_g );

**PARAMETER:** INPUT LOGS \*gs Log handle collection

INPUT long \*p\_index Statement for the first time a long type variables are initialized to LOG\_TRAVELLOG\_INDEX\_INIT handed in, as the traverse the tracking number

OUTPUT char \*\*pp\_g\_id Every time after the success of the traversal set logging handle identifier string pointer

OUTPUT LOG \*\*pp\_g Every time after the success of the traversal set logging handle pointer

**RETURN:** Successful, return 0

Traverse over, return to LOGS\_RETURN\_INFO\_NOTFOUND

Fails, the return to the other

**REMARK:**

## 4 Configuration file interface layer functions

### 4.1 Log handle

#### 4.1.1 Handle to the build log from the config file

**PROTOTYPE:** LOG \*CreateLogHandleFromConfig( char \*config\_filename , char \*postfix );

**PARAMETER:** INPUT The final configuration file names can be config\_filename, also can by config\_filename and postfix patchwork

**RETURN:** Successful, returns a handle to the newly created log  
Failure, returns NULL, general memory failure for application

**REMARK:**

### 4.2 Log handle collection

#### 4.2.1 The build log handle collection from the config file

**PROTOTYPE:** LOGS \*CreateLogsHandleFromConfig( char \*config\_filename , char \*postfix );

**PARAMETER:** INPUT The final configuration file names can be config\_filename, also can by config\_filename and postfix patchwork

**RETURN:** Successful, returns a handle to the newly created log collection  
Failure, returns NULL, general memory failure for application

**REMARK:**

## 5 Configuration of auxiliary function

### 5.1 Property transfer function

#### 5.1.1 Log output type (string converted to an integer)

**PROTOTYPE:** int ConvertLogOutput\_atoi( char \*output\_desc , int \*p\_log\_output );

**PARAMETER:** INPUT char \*output\_desc log output description

OUTPUT int \*p\_log\_output log output

**RETURN:** Successful, return 0

Failure, returns non-zero, usually is caused by the input parameter is invalid

**REMARK:**

#### 5.1.2 Log level types (string converted to an integer)

**PROTOTYPE:** int ConvertLogLevel\_atoi( char \*log\_level\_desc , int \*p\_log\_level );

**PARAMETER:** INPUT char \*log\_level\_desc log level description

OUTPUT int \*p\_log\_level log level

**RETURN:** Successful, return 0

Failure, returns non-zero, usually is caused by the input parameter is invalid

**REMARK:**

#### 5.1.3 Log level types (integer is converted to a string)

**PROTOTYPE:** int ConvertLogLevel\_itoa( int log\_level , char \*\*log\_level\_desc );

**PARAMETER:** INPUT int log\_level\_desc

OUTPUT char \*\*log\_level\_desc

**RETURN:** Successful, return 0

Failure, returns non-zero, usually is caused by the input parameter is invalid



**REMARK:**

### 5.1.4 Line log style (string converted to an integer)

**PROTOTYPE:** `int ConvertLogStyle_atol( char *line_style_desc , long *p_line_style );`

**PARAMETER:** INPUT `char *line_style_desc`

OUTPUT `long *p_line_style`

**RETURN:** Successful, return 0

Failure, returns non-zero, usually is caused by the input parameter is invalid

**REMARK:** This function only convert a single style values

### 5.1.5 Log option (string converted to an integer)

**PROTOTYPE:** `int ConvertLogOption_atol( char *log_option_desc , long *p_log_option );`

**PARAMETER:** INPUT `char *log_option_desc`

OUTPUT `long *p_log_option`

**RETURN:** Successful, return 0

Failure, returns non-zero, usually is caused by the input parameter is invalid

**REMARK:** This function is only convert a single option value

### 5.1.6 Log rolling mode (string is converted to an integer)

**PROTOTYPE:** `int ConvertLogRotateMode_atoi( char *rotate_mode_desc , int *p_rotate_mode );`

**PARAMETER:** INPUT `char *rotate_mode_desc`

OUTPUT `int *p_rotate_mode`

**RETURN:** Successful, return 0

Failure, returns non-zero, usually is caused by the input parameter is invalid

**REMARK:**

### 5.1.7 The log buffer (string converted to an integer)

**PROTOTYPE:** int ConvertBufferSize\_atol( char \*bufsize\_desc , long \*p\_bufsize );

**PARAMETER:** INPUT char \*bufsize\_desc

OUTPUT long \*p\_bufsize

**RETURN:** Successful, return 0

Failure, returns non-zero, usually is caused by the input parameter is invalid

**REMARK:** Support unit suffix: "B"、"KB"、"MB"、"GB"

## 6 Simple configuration file attributes list

ATTRIBUTE	NOTE
output log_level [ filename ]	
level log_level	
custlabel1~custlabel3 custom_label	custom label 1~3
styles line_log_style	Before and after use ' ' combination, use double quotation marks
options log_options	Before and after use ' ' combination, enclosed in double quotation marks; Don't set the default to CHANGE_TEST
rotate_mode log_rolling_mode	
rotate_size log_rolling_size	Support unit suffix: "GB"、"MB"、"KB"
rotate_pressure_factor factor	
log_buffersize bufsize [max_bufsize ]	Support unit suffix
hexlog_buffersize bufsize [max_bufsize ]	Support unit suffix

Sample

id	hello
----	-------

output	FILE test_logconf.log
level	INFO
styles	DATETIME LOGLEVEL PID TID SOURCE FORMAT NEWLINE
options	CHANGE_TEST
rotate_mode	SIZE
rotate_size	10MB
log_bufsize	1MB 5MB

or

id	hello
output	FILE test_logconf.log
level	INFO
styles	DATETIME LOGLEVEL PID TID SOURCE FORMAT NEWLINE
options	CHANGE_TEST
rotate_mode	SIZE
rotate_size	10MB
log_bufsize	1MB 5MB
hexlog_bufsize	5MB

  

id	stdout
output	STDOUT
level	INFO
styles	DATETIME LOGLEVEL PID TID SOURCE FORMAT NEWLINE