MDLOGGER configuration documentation

Rev. 1.0.0

MDLOGGER is configured using a classic **.ini** stlyle file.
There must be a mandatory **[GLOBAL]** section plus a number of section one for each log handler (minimum 1 section for the root hanlder).

GLOBAL section

This section is mandatory, it contains the following keys:

Key	Type	Note
enabled	Boolean (true or false)	Mandatory otherwise mdlogger does not start
pattern	Text	Missing pattern key means pattern = [%{timestamp:utc} %{msg_type}] %{message}
Valid pattern placeholder		
%{timestamp:utc}		**log message utc time timestamp**
%{timestamp:loc}		**log message local time timestamp**
%{msg_type}		**log message type**
%{appname}		**application name**
%{appversion}		**application version**
%{thread}		**thread name or thread id if name has not been set where the log has been made**
%{category}		**log message category**
%{file}		**file name where the log has been made**
%{function}		**function name where the log has been make**
%{line}		**source line number where the log has been made**
%{message}		**user message**
timestamp_format	Text	1) if timestamp_format missing the logger does not start. 2) if it is empty the default '[year]-[month]-[day] [hour]:[minute]:[second].[subsecond digits:3] [offset_hour sign:mandatory]:[offset_second]' is used. Due to the use of '[' & ']" fotmat has to be sorround by ''
debug.enabled	Boolean (true or	missing message type enabling flag does

Key	Type	Note
	false)	not make start the mdlogger
info.enabled	Boolean (true or false)	missing message type enabling flag does not make start the mdlogger
warnig.enabled	Boolean (true or false)	missing message type enabling flag does not make start the mdlogger
critical.enabled	Boolean (true or false)	missing message type enabling flag does not make start the mdlogger
fatal.enabled is not managed, fatal messages cannot be disabled		
debug.text	Text	missing message type Text key does not make start the mdlogger, if it's empty then the fallowing value is assumed: Debug
info.text	Text	missing message type Text key does not make start the mdlogger, if it's empty then the fallowing value is assumed: Info
warning.text	Text	missing message type Text key does not make start the mdlogger, if it's empty then the fallowing value is assumed: Warning
critical.text	Text	missing message type Text key does not make start the mdlogger, if it's empty then the fallowing value is assumed: Critical
fatal.text	Text	missing message type Text key does not make start the mdlogger, if it's empty then the fallowing value is assumed: Fatal
root_log_handler	Text	empty or missing value does not make start the mdlogger
external_command.ipaddress	Text valid IPV4 address (e.g. 192.2.1.183) or IPV6 address (e.g fe80::61fb:38bd:9266:52e8	if external commands ip address are missing or wrong a warning messag is printed out an no commands will be managed ip address could be unicast or multicast (IPV4 or IPV6) and you can specify on wich interface multicast using external_command.multicast_if i.g external_command.multicast_if = 192.168.207.128 otherwise multicast message will be

Key	Type	Note
		received on all network interfaces
external_command.multicast_if	IPV4 or IPV6 address format	Use this key only in case the external_command.ipaddress key contains a multicast address **maintain consistency between the IP address type of the external_command.ipaddress and external_command.multicast_if key**
external_command.port	Positive no 0 integer value IP port with range 1-65535	cannot be 0

Example:

```
[GLOBAL]

enabled = true

pattern = [%{timestamp:loc}> %{msg_type}, %{thread}, %{category}, %{function}, %{line}] %{message}

timestamp_format = "[year]-[month]-[day] [hour]:[minute]:[second].[subsecond digits:3]

debug.enabled = true

info.enabled = true

warning.enabled = true

critical.enabled = true

debug.text = D

info.text = I

warning.text = W

critical.text = C

fatal.text = F

root_log_handler = CONSOLE

external_command.ipaddress = 192.168.207.128

external_command.port = 54321
```

Predefined log handlers

There are 4 predefined log handler types

1) console

2) file (rolling file)

3) network

4) unix-doman (unix domain socket, udp/tcp, **windows os manages tcp only**)

The following global keys can be overwritten by all predefined log handlers

see **get_log_handler_common_parameters** in the source code of **utils module**

pattern
timestamp_format
debug.enabled
info.enabled
warning.enabled
critical.enabled
debug.text
info.text
warning.text
critical.text
fatal.text

Console log handler

This kind of log handler is able to print out application logs on the process standard output and/or standard error file descriptor.

For each log message type (debug, info, warning, critical and fatal) you can chose where messages will be printed out.

Keys	Type	Note
type	Fixed Text **console**	
enabled	Boolean (true or false)	missing enabled key means enabled = false
log_message_format	Text, valid values are: 1) plain_Text 2) json 3) json_pretty	missing log_message_format means that plain_Text format will be used
debug.redirection	Text, valid values are: 1) stdout 2) stderr 3 both	missing redirection causes the creation of log handler to fail
info.redirection	Text, valid values are: 1) stdout 2) stderr 3 both	missing redirection causes the creation of log handler to fail
warning.redirection	Text, valid values are: 1) stdout 2) stderr 3 both	missing redirection causes the creation of log handler to fail
critical.redirection	Text, valid values are: 1) stdout 2) stderr 3 both	missing redirection causes the creation of log handler to fail
fatal.redirection	Text, valid values are: 1) stdout 2) stderr 3 both	missing redirection causes the creation of log handler to fail

File log handler

This type of log handler writes log message to a series of files (rolling file mechanism) at least one file only.

Keys	Type	Note
type	Fixed Text **file**	
enabled	Boolean (true or false)	missing enabled key means enabled = false
log_message_format	Text, valid values are: 1) plain_Text 2) json 3) json_pretty	missing log_message_format means that plain_Text format will be used
remove_previous_logs	Boolean (true or false)	missing or empty remove_previous_logs is assumed = true. This flag enables or disables the possibility to remove previous logging file contained in the log directory
directory	Text	missing or empty remove_previous_logs is assumed to be process current working directory
basename	Text	could be any not empty value or you can use specific placeholders

%{appname}	**application name as passed to initialize function**
%{datetime:utc}	**start logging date**
%{datetime:loc}	**start logging local time**

datetime format is [year][month][day]_[hour][minute][second]_[offset_hour][offset_second]"

Keys	**Type**	**Note**
extension	Text	missing or empty extension is allowed (dot is automatically inserted if missed)
maxsize	Positive integer value+unit measure enumeration value (e.g. 200MB)	size cannot be missed or less than or equal to 0
B	**Bytes**	
KB	**Kilo bytes**	
MB	**Mega bytes**	
GB	**Giga bytes**	
TB	**Tera bytes**	
depth	Positive integer number	has to be greater or equal to 1 and if missed is assumed = 3

Network log handler

This kind o log handler sends application log messages over the network through a socket

Valid network protocols are: **udp unicast, udp multicast, tcp.**

IP address can be **IPV4** or **IPV6**.

Keys	Type	Note
type	Fixed Text **network**	
enabled	Boolean (true or false)	missing enabled key means enabled = false
log_message_format	Text, valid values are: 1) plain_Text 2) json 3) json_pretty	missing log_message_format means that plain_Text format will be used
protocol	Text valid value are: 1) udp 2) mcast 3) tcp	cannot be missed or empty otherwise it causes the creation of log handler to fail
remote_address	Text valid IPV4 address (e.g. 192.2.1.183) or IPV6 address (e.g fe80::61fb:38bd:9266:52e8)	cannot be missed or empty otherwise it causes the creation of log handler to fail
remote_port	Positive no 0 integer value IP port with range 1-65535	cannot be missed or empty or 0 otherwise it causes the creation of log handler to fail
multicast_if	Text valid IPV4 address (e.g. 192.2.1.183) or IPV6 address (e.g fe80::61fb:38bd:9266:52e8)	Use this key only in case the log handler remote_address key contains a multicast address **maintain consistency between the IP address type of the remote_address key and the multicast_if key**

Unix-domain socket log handler

This kind o log handler sends application log messages through an unix-domin socket.

Note: On windows OS you can use tcp protocol only, udp is not allowed

Keys	Type	Note
type	Fixed Text **network**	
enabled	Boolean (true or false)	missing enabled key means enabled = false
log_message_format	Text, valid values are: 1) plain_Text 2) json 3) json_pretty	missing log_message_format means that plain_Text format will be used
protocol	Text valid value are: 1) udp (not on windows os) 3) tcp	cannot be missed or empty otherwise it causes the creation of log handler to fail
sun_path	Text valid unix-domain path (e.g /tmp/local_server) sun_path can contains environment variable (espressed in unix style ${<varname>}) or an absolute path which size cannot be greater than 108 bytes (e.g sun_path = ${TMP}\local_server)	cannot be missed or empty otherwise it causes the creation of log handler to fail