MDLOGGER external commands

Rev. 1.0.0

As you can read in the documentation of MDLOGGER configuration file it is possible for the logger to receive external command to modify at run time its configuration (GLOBAL or of a spcific log handler)

Commands are text messages expressed in json format that is the following

```
{
   "command": "<valid command name>",
   "parameters": []
}
```

Accepted commands are

get-config

set-global

set-handler

paramters field is an array of json objects that have the following format

```
{
        “name”: “<parameter name>”,
        “value”: <json value>
}
```

parameters cab be an empty array or a json value according the command

when the logger receive a command an answer is replayed. The answer to the command has a json format too.

```
{
        "ack_nack": "<ack_nack value>",
        “reason”: “<reason text”,
        “value”:  <json value> (Any possible json value type)
}
```

"ack_nack" could be: **“ACK”, “NACK”, “PARTIALACK”**

A partial ack could happen for example when the changing of a setting value successfully occured but there was a problem to save it in the configuration file.

This is an example to a get-config command

COMMAND

```
 {
  "command": "get-config",
  "parameters": []
}
```

ANSWER

```
{
  "ack_nack": "ACK",
  "reason": "",
  "value": {
    "global": {
      "critical.enabled": true,
      "critical.text": "C",
      "debug.enabled": true,
      "debug.text": "D",
      "enabled": true,
      "external_command.ipaddress": "192.168.1.183",
      "external_command.port": 54321,
      "fatal.text": "F",
      "info.enabled": true,
      "info.text": "I",
      "pattern": "[<%{timestamp:utc} | %{timestamp:loc}> %{msg_type}, %{appname}, %{appvarsion}, %
{thread}, %{category}, %{file}, %{function}, %{line}] %{message}",
      "root_log_handler": "CONSOLE",
      "timestamp_format": "\"[year]-[month]-[day] [hour]:[minute]:[second].[subsecond digits:3]
[offset_hour sign:mandatory]:[offset_second]\"",
      "warning.enabled": true,
    "warning.text": "W"
  },
  "log_handlers": [
    {
      "appname": "mdlogger_test",
      "appver": "1.0.0",
      "critical.enabled": true,
      "critical.redirection": "stdout",
      "critical.text": "C",
      "debug.enabled": true,
      "debug.redirection": "stdout",
      "debug.text": "D",
      "enabled": true,
      "fatal.eanbled": true,
      "fatal.redirection": "stdout",
      "fatal.text": "F",
      "info.enabled": true,
      "info.redirection": "stdout",
      "info.text": "I",
      "log_message_format": "plain_text",
      "name": "CONSOLE",
      "pattern": "[<%{timestamp:utc} | %{timestamp:loc}> %{msg_type}, %{appname}, %{appvarsion}, %
{thread}, %{category}, %{file}, %{function}, %{line}] %{message}",
      "timestamp_format": "\"[year]-[month]-[day] [hour]:[minute]:[second].[subsecond digits:3]
[offset_hour sign:mandatory]:[offset_second]\"",
      "warning.enabled": true,
      "warning.redirection": "stdout",
      "warning.text": "W"
    }
  ]
 }
}
```

Example of set-global command

COMMAND

```
{
  "command": "set-global",
  "parameters": [
    {
      "name": "enabled",
      "value": false
    }
  ]
}
```

ANSWER

```
{
        "ack_nack":"ACK",
        "reason":"enabled changed and saved",
        "value":null
}
```

When a set-global command is performed correctly, the new value is automatically saved in the configuration file for the next running.

When a set-handler command is performed corrctly on a predefined log handler you can choose to save it or not aading the save parameter.

Example

COMMAND

```
{
  "command": "set-handler",
  "parameters": [
    {
      "name": "log_handler",
      "value": "CONSOLE"
    },
    {
      "name": "key",
      "value": "enabled"
    },
    {
      "name": "new-value",
      "value": true
    },
    {
      "name": "save",
      "value": true
    }
  ]
}
```

ANSWER

```
{
            "ack_nack":"ACK",
            "reason":"Log handler \'CONSOLE\' parameter \'enabled\' changed and saved for the next run",
            "value":null
}
```