

-Handler

v2.0glogHandlerHandlerHandlerHandler

Handler

```
// Handler is function handler for custom logging content outputs.
type Handler func(ctx context.Context, in *HandlerInput)
```

HandlerHandler

Handler

```
// HandlerInput is the input parameter struct for logging Handler.
type HandlerInput struct {
    Logger      *Logger      // Current Logger object.
    Buffer       *bytes.Buffer // Buffer for logging content outputs.
    Time        time.Time    // Logging time, which is the time that
logging triggers.
    TimeFormat  string       // Formatted time string, like "2016-01-
09 12:00:00".
    Color       int          // Using color, like COLOR_RED,
COLOR_BLUE, etc. Eg: 34
    Level       int          // Using level, like LEVEL_INFO,
LEVEL_ERROR, etc. Eg: 256
    LevelFormat string       // Formatted level string, like "DEBU",
"ERROR", etc. Eg: ERROR
    CallerFunc  string       // The source function name that calls
logging, only available if F_CALLER_FN set.
    CallerPath  string       // The source file path and its line
number that calls logging, only available if F_FILE_SHORT or F_FILE_LONG
set.
    CtxStr      string       // The retrieved context value string
from context, only available if Config.CtxKeys configured.
    TraceId     string       // Trace id, only available if
OpenTelemetry is enabled.
    Prefix      string       // Custom prefix string for logging
content.
    Content     string       // Content is the main logging content
without error stack string produced by logger.
    Values      []any        // The passed un-formatted values array
to logger.
    Stack       string       // Stack string produced by logger, only
available if Config.StStatus configured.
    IsAsync     bool        // IsAsync marks it is in asynchronous
logging.
}
```

Handler

- HandlerInputin.Next()HandlerInput
- BufferBuffer

HandlerLogger

```
// SetHandlers sets the logging handlers for current logger.
func (l *Logger) SetHandlers(handlers ...Handler)
```

Content Menu

- - [Handler](#)
 - [Handler](#)
 - [HandlerLogger](#)
- - [1. Json](#)
 - [2.](#)
- [Handler](#)
- [Handler](#)
 - [HandlerJson](#)
 - [HandlerStructure](#)

Handler

1. Json

HandlerJSON

```
package main

import (
    "context"
    "encoding/json"
    "os"

    "github.com/gogf/gf/v2/frame/g"
    "github.com/gogf/gf/v2/os/glog"
    "github.com/gogf/gf/v2/text/gstr"
)

// JsonOutputsForLogger is for JSON marshaling in sequence.
type JsonOutputsForLogger struct {
    Time    string `json:"time"`
    Level   string `json:"level"`
    Content string `json:"content"`
}

// LoggingJsonHandler is a example handler for logging JSON format content.
var LoggingJsonHandler glog.Handler = func(ctx context.Context, in *glog.
HandlerInput) {
    jsonForLogger := JsonOutputsForLogger{
        Time:    in.TimeFormat,
        Level:    gstr.Trim(in.LevelFormat, "[]"),
        Content:  gstr.Trim(in.Content),
    }
    jsonBytes, err := json.Marshal(jsonForLogger)
    if err != nil {
        _, _ = os.Stderr.WriteString(err.Error())
        return
    }
    in.Buffer.Write(jsonBytes)
    in.Buffer.WriteString("\n")
    in.Next(ctx)
}

func main() {
    g.Log().SetHandlers(LoggingJsonHandler)
    ctx := context.TODO()
    g.Log().Debug(ctx, "Debugging...")
    g.Log().Warning(ctx, "It is warning info")
    g.Log().Error(ctx, "Error occurs, please have a check")
}
```

HandlerBufferHandlerBufferNextHandler

```
{"time":"2021-12-31 11:03:25.438","level":"DEBU","content":"Debugging..."}
{"time":"2021-12-31 11:03:25.438","level":"WARN","content":"It is warning
info"}
{"time":"2021-12-31 11:03:25.438","level":"ERROR","content":"Error occurs,
please have a check \nStack:\n1.  main.main\n    C:/hailaz/test/main.go:
42"}
```

2.

Handlergraylog

```
package main

import (
    "context"

    "github.com/gogf/gf/v2/frame/g"
    "github.com/gogf/gf/v2/os/glog"
    gelf "github.com/robertkowalski/graylog-golang"
)

var grayLogClient = gelf.New(gelf.Config{
    GraylogPort:      80,
    GraylogHostname: "graylog-host.com",
    Connection:       "wan",
    MaxChunkSizeWan: 42,
    MaxChunkSizeLan: 1337,
})

// LoggingGrayLogHandler is an example handler for logging content to
// remote GrayLog service.
var LoggingGrayLogHandler glog.Handler = func(ctx context.Context, in
*glog.HandlerInput) {
    in.Next()
    grayLogClient.Log(in.Buffer.String())
}

func main() {
    g.Log().SetHandlers(LoggingGrayLogHandler)
    ctx := context.TODO()
    g.Log().Debug(ctx, "Debugging...")
    g.Log().Warning(ctx, "It is warning info")
    g.Log().Error(ctx, "Error occurs, please have a check")
    glog.Print(ctx, "test log")
}
```

Handler

Handlerv2.1HandlerHandlerHandlerHandler

Handler

```
// SetDefaultHandler sets default handler for package.
func SetDefaultHandler(handler Handler)

// GetDefaultHandler returns the default handler of package.
func GetDefaultHandler() Handler
```



JSON

```

package main

import (
    "context"
    "encoding/json"
    "os"

    "github.com/gogf/gf/v2/frame/g"
    "github.com/gogf/gf/v2/os/glog"
    "github.com/gogf/gf/v2/text/gstr"
)

// JsonOutputsForLogger is for JSON marshaling in sequence.
type JsonOutputsForLogger struct {
    Time    string `json:"time"`
    Level   string `json:"level"`
    Content string `json:"content"`
}

// LoggingJsonHandler is a example handler for logging JSON format content.
var LoggingJsonHandler glog.Handler = func(ctx context.Context, in *glog.
HandlerInput) {
    jsonForLogger := JsonOutputsForLogger{
        Time:    in.TimeFormat,
        Level:    gstr.Trim(in.LevelFormat, "["],"),
        Content:  gstr.Trim(in.Content),
    }
    jsonBytes, err := json.Marshal(jsonForLogger)
    if err != nil {
        _, _ = os.Stderr.WriteString(err.Error())
        return
    }
    in.Buffer.Write(jsonBytes)
    in.Buffer.WriteString("\n")
    in.Next(ctx)
}

func main() {
    ctx := context.TODO()
    glog.SetDefaultHandler(LoggingJsonHandler)

    g.Log().Debug(ctx, "Debugging...")
    glog.Warning(ctx, "It is warning info")
    glog.Error(ctx, "Error occurs, please have a check")
}

```

```

{"time":"2022-06-20 10:51:50.235","level":"DEBU","content":"Debugging..."}
{"time":"2022-06-20 10:51:50.235","level":"WARN","content":"It is warning
info"}
{"time":"2022-06-20 10:51:50.235","level":"ERRO","content":"Error occurs,
please have a check"}

```

Handler

Handler

HandlerJson

HandlerJson

```

package main

import (
    "context"

    "github.com/gogf/gf/v2/frame/g"
    "github.com/gogf/gf/v2/os/glog"
)

func main() {
    ctx := context.TODO()
    glog.SetDefaultHandler(glog.HandlerJson)

    g.Log().Debug(ctx, "Debugging...")
    glog.Warning(ctx, "It is warning info")
    glog.Error(ctx, "Error occurs, please have a check")
}

```

```

{"Time":"2022-06-20 20:04:04.725","Level":"DEBU","Content":"Debugging..."}
{"Time":"2022-06-20 20:04:04.725","Level":"WARN","Content":"It is warning info"}
{"Time":"2022-06-20 20:04:04.725","Level":"ERROR","Content":"Error occurs, please have a check","Stack":"1. main.main\n    /Users/john/Workspace/Go/GOPATH/src/github.com/gogf/gf/.test/main.go:16\n"}

```

HandlerStructure

HandlerGolangslog

```

package main

import (
    "context"
    "net"

    "github.com/gogf/gf/v2/frame/g"
    "github.com/gogf/gf/v2/os/glog"
)

func main() {
    ctx := context.TODO()
    glog.SetDefaultHandler(glog.HandlerStructure)

    g.Log().Info(ctx, "caution", "name", "admin")
    glog.Error(ctx, "oops", net.ErrClosed, "status", 500)
}

```

```

Time="2023-11-23 21:00:08.671" Level=INFO Content=caution name=admin
Time="2023-11-23 21:00:08.671" Level=ERROR oops="use of closed network connection" status=500 Stack="1. main.main\n    /Users/txqiangguo/Workspace/gogf/gf/example/.test/main.go:16\n"

```