# SFLOG_LogMsg

Last Modified on 01/18/2017 10:23 pm CST

- C/C++
- .Net

```
int __stdcall SFLOG_LogMsg(int          nLevel,
                           LPCTSTR      szFilename,
                           LPCTSTR      szFuncName,
                           LPCTSTR      szFuncSig,
                           int          nLineNo,
                           LPCTSTR      szMsg
                          )
```

log a message into the logging system

**Returns**

status code of the operation

**Return values**

**NDK_SUCCESS** Operation successful

**NDK_FAILED** Operation unsuccessful. See **Macros** for full list.

**Parameters**

**nLevel** logging level (i.e. debug, info, trace, error, etc.).

| Level | Macro | Description |
|---|---|---|
| 1 | SFLOG_TRACE | Trace level logging |
| 2 | SFLOG_DEBUG | Debug level logging |
| 3 | SFLOG_INFO | Information level logging |
| 4 | SFLOG_WARN | Warning level logging |
| 5 | SFLOG_ERROR | Error level logging |
| 6 | SFLOG_FATAL | Fatal or critical error logging |

**szFilename** the source filename that triggers this logging message

**szFuncName** the function name from which this log is triggered from

**szFuncSig** the function sugnature (i.e. mangled name)

**nLineNo** Line number in the source file

**szMsg** Error message

**Remarks**

- This function will fail, and return (NDK_LOG_UNINITIALIZED), if the logging system has not been initialized yet.
- C/C++ compiler has a set of standard predefined preprocessors that can be used when calling this function:
  - __FILE__ : This macro expands to the name of the current input file

- __LINE__ : This macro expands to the current input line number, in the form of a decimal integer constant
- Microsoft C/C++ compiler offers additional predefined macros that we can use when calling this function:
  - __FUNCTION__ : This macro expands to the the undecorated name of the enclosing function as a string literal
  - __FUNCSIG__ : This macro expands to the the signature of the enclosing function as a string literal
- For convenience, you may wish to define few macros to automate the logging further. For example:

```
#define LOG_INFO(x) \
SFLOG_logMsg(SFLOG_INFO, __FILE__,__FUNCTION__,__FUNCSIG__,__LINE__,x);

#define LOG_ERROR(x) \
SFLOG_logMsg(SFLOG_ERROR, __FILE__,__FUNCTION__,__FUNCSIG__,__LINE__,x);

#define LOG_WARN(x) \
SFLOG_logMsg(SFLOG_ERROR, __FILE__,__FUNCTION__,__FUNCSIG__,__LINE__,x);
```

**Requirements**

| | |
|---|---|
| **Header** | SFLogger.H |
| **Library** | SFLOG.LIB |
| **DLL** | SFLOG.DLL |

**Examples**

```
#define LOG_ERROR(x)SFLOG_logMsg(SFLOG_ERROR, __FILE__,__FUNCTION__,__FUNCSIG__,_
_LINE__,x)
....
nRet = NDK_SESMTH( ...);
if(nRet < NDK_SUCCESS){
LOG_ERROR("NDK_DESMTH failed!");

...
}
```

```
NDK_RETCODE LogMsg(int      nLevel,
                   string   szFilename,
                   string   szFuncName,
                   string   szFuncSig,
                   int      nLineNo,
                   string   szMsg,
                  )
```

**Namespace:** NumXLAPI
**Class:** SLOG
**Scope:** Public
**Lifetime:** Static

log a message into the logging system

**Return Value**

a value from **NDK_RETCODE** enumeration for the status of the call.

**NDK_SUCCESS** operation successful
Error Error Code

**Parameters**

[in]**nLevel** logging level (i.e. debug, info, trace, error, etc.).

| Level | Macro | Description |
|---|---|---|
| 1 | SFLOG_TRACE | Trace level logging |
| 2 | SFLOG_DEBUG | Debug level logging |
| 3 | SFLOG_INFO | Information level logging |
| 4 | SFLOG_WARN | Warning level logging |
| 5 | SFLOG_ERROR | Error level logging |
| 6 | SFLOG_FATAL | Fatal or critical error logging |

[in]**szFilename** the source filename that triggers this logging message
[in]**szFuncName** the function name from which this log is triggered from
[in]**szFuncSig** the function sugnature (i.e. mangled name)
[in]**nLineNo** Line number in the source file
[in]**szMsg** Error message

**Remarks**

- This function will fail, and return (NDK_LOG_UNINITIALIZED), if the logging system has not been initialized yet.
- To automate the logging process further, we recommend defining the following function:

```
public static void LogMessage(SFLOG_LEVEL nLevel, string message,
[System.Runtime.CompilerServices.CallerMemberName] string memberName = "",
[System.Runtime.CompilerServices.CallerFilePath] string sourceFilePath = "
",
[System.Runtime.CompilerServices.CallerLineNumber] int sourceLineNumber =
0)
{

SFLOG.LogMsg(nLevel, sourceFilePath, memberName, memberName, sourceLineNum
ber, message);
}
```

Where the .Net compiler will auto-fill the function name, filename, etc.

**Exceptions**

| Exception Type | Condition |
|---|---|
| None | N/A |

**Requirements**

| | |
|---|---|
| **Namespace** | NumXLAPI |
| **Class** | SFLOG |
| **Scope** | Public |
| **Lifetime** | Static |
| **Package** | NumXLAPI.DLL |

**Examples**

```
....
}
catch (Exception exc)
{
LogMessage(SFLOG_LEVEL.SFLOG_FATAL, "Failed to establish a database connection");
LogMessage(SFLOG_LEVEL.SFLOG_FATAL, exc.ToString());
```

## References


* Hamilton, J .D.; Time Series Analysis , Princeton University Press (1994), ISBN 0-691-04289-6
* Tsay, Ruey S.; Analysis of Financial Time Series John Wiley & SONS. (2005), ISBN 0-471-690740
* D. S.G. Pollock; Handbook of Time Series Analysis, Signal Processing, and Dynamics; Academic Press; Har/Cdr edition(Nov 17, 1999), ISBN: 125609906
* Box, Jenkins and Reisel; Time Series Analysis: Forecasting and Control; John Wiley & SONS.; 4th edition(Jun 30, 2008), ISBN: 470272848


## See Also

[template("related")]