SFLOG_Init

Last Modified on 01/18/2017 10:22 pm CST

- C/C++
- .Net

```
int __stdcall SFLOG_Init(LPCTSTR szAppName,
                         LPCTSTR szLogDir
                        )
```

This routine initialize the Logging system. This is often the first API call that an application makes and is a prerequisite for other logging APIs.

Parameters

[in]	**szAppName**	is the application name (user-defined), but must match the configuration base filename
[in]	**szLogDir**	is the directory path to use for storing logging file and temporary files.

Returns

status code of the operation

Return values

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

Remarks

- The value of the application name argument (i.e. szAppName) must match the name of the configuration file. The confguration file must exist in the same folder as your application executable file (e.g. MyApp.exe)
- If the value of szLogDir is missing (empty or NULL), the function will use the default temp directory in the current user's profile.
- The logging system uses reference count to manage the system lifetime and support multiple clients to obtain and release access to the system without conditioning on one another in managing the system lifetime.
- For custom application, if you use **NDK_Init**, then you don't need to invoke this function in your application. The **NDK_init** will initialized the logging system on your behalf.
- For multiple concurrent running processes (e.g. custom application, NumXL Add-in, etc.), the logging system will open/create a separate log file (with a unique suffix) for each process.

Requirements

Header	SFLogger.H
Library	SFLOG.LIB
DLL	SFLOG.DLL

Examples

```
#include
#include

// Link with SFLOG.lib
#pragma comment("lib", "SFLOG.lib")

using std;

void main(void)
{
int nRet= NDK_FAILED;

string szAppName="MyLogExample";
string szPath = "C:\\temp";

nRet = SFLOG_Init( szAppName.c_str(), // Application name (used for log file
name (e.g. MyLogExample.log)).
szPath.c_str()); // log directory where log files are created
if( nRet >= NDK_SUCCESS)
{
// Is the log system initialization OK?
```

NDK_RETCODE Init(string szAppName,
string szLogDir
)

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

This routine initialize the Logging system. This is often the first API call that an application makes and is a prerequisite for other logging APIs.

Parameters

[in] **szAppName** Application name (e.g. TestApp).
[in] **szLogDir** Temporary directory to use for storing logging file and intermediate files.

Return Value

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

NDK_SUCCESS operation successful
Error Error Code

Remarks

- The value of the application name argument (i.e. szAppName) must match the name of the configuration file. The confguration file must exist in the same folder as your application executable file (e.g. MyApp.exe)
- If the value of szLogDir is missing (empty or NULL), the function will use the default temp directory in the current user's profile.
- The logging system uses reference count to manage the system lifetime and support multiple clients to obtain and release access to the system without conditioning on one another in managing the system lifetime.
- For custom application, if you use **SFDK.Init**, then you don't need to invoke this function in your application. The **SFSDK.init** will initialized the logging system on your behalf.
- For multiple concurrent running processes (e.g. custom application, NumXL Add-in, etc.), the logging system will open/create a separate log file (with a unique suffix) for each process.

Exceptions

Exception Type	Condition
None	N/A

Requirements

Namespace	NumXLAPI
Class	SFLOG
Scope	Public
Lifetime	Static
Package	NumXLAPI.DLL

Examples

```
using NumXLAPI;
...
...
NDK_RETCODE retCode = NDK_RETCODE.NDK_FAILED;

string szAppName = "RastPro";
retCode = NumXLAPI.SFLOG.Init(szAppName, null);
if (retCode
```

See Also

[template("related")]