KIWI ENTERPRISES

Quality networking tools at affordable prices

Kiwi Logger (KLOG) v2.0

Adds syslogging functionality to your application

by Kiwi Enterprises

Kiwi Logger (KLOG) is a suite of syslog message sending tools for the Windows platform. The tools allow you to add syslog message sending capabilities to your applications. The syslog messages can be received by Kiwi Syslog Daemon (or compatible).

You can access KLOG via console command line, Windows command line, DLL API or ActiveX component.

Table of Contents

1 Introduction

KIWI ENTERPRISES
Quality networking tools at affordable prices

Kiwi Logger (KLOG) v2.0

Copyright 2002-2006, with all rights reserved by Kiwi Enterprises.

Kiwi Logger (KLOG) is a suite of syslog message sending tools for the Windows platform. The tools allow you to add syslog message sending capabilities to your applications. The syslog messages can be received by Kiwi Syslog Daemon (or compatible).

You can access KLOG via console command line, Windows command line, or leverage the SyslogSender API Windows DLL, COM DLL, or .NET DLL.

Integration examples are included for command line, C, VC++, Visual Basic, ASP, VBScript, ASP.net and VB.net.
The DLL can also be dynamically or statically linked via the included library file.

These DLLs, libraries, samples and executables are provided as freeware.

For information on component redistribution please see License Agreement

Kiwi Enterprises Website: http://www.kiwisyslog.com

2 Product components

The following products are available in the KLOG suite:

KLOG (exe)
The console-mode syslog message sender. See the Usage section for its operation. This is the original product in the suite.

KLOGWIN (exe)
A Windows GUI version of the console-mode utility. It does not run from a console, so it is more suitable for being called from Windows GUI applications (no console window is opened). It uses the same program arguments as the console-mode version, except it cannot repeat messages every second.

KLOG.DLL
This DLL provides KLOG functionality to other applications. It is used by the KSEND

sample application. Refer to the *Using KLOG functions from other Windows applications* section.

KSEND (exe/sample)
A Windows GUI tool for sending messages written in C++. Its interface provides the same options as KLOGWIN's command line. It can send one or ten messages. It is designed as a test utility and can not be called by other applications. KSend uses the KLOG.DLL to send messages.

KLOG.LIB
The import library which can be used by developers. By linking with KLOG.LIB, KLOG.DLL features can be leveraged in other applications.

KLOG_COM.dll (COM)
Suitable for use with COM-compliant languages such as VB6, ASP, and VBScript. Examples included in the ../KLOGDLL_COM/Samples directory illustrate the COM component usage in VB6, ASP and VBScript.

KLOG_NET.dll (.net)
A .net Syslog Sender library. Samples included in ../KLOGDLL_NET/Samples. "KlogWin" is a VB.net example of a .net syslog sender leveraging the KLOG_NET.dll. "KlogWeb" is a ASP.net sample syslog sending application very similar to KLogWin.

3 Usage

This section describes the command line options for the console-mode and Windows executable. All other products in the suite share the functionality.

From the command line, run **klog.exe** (or klogwin.exe) with the command line switches required for your message. To display copyright details use **klog -c**. For syntax help, use **klog -?**. For facility and level names, use **klog –d**.

The following command line switches are recognised by Kiwi Logger:

Flag	Description
-u <port>	Destination port on the Syslog Daemon host
-h <host>	Destination host address of the Syslog Daemon
-p <priority>	Message priority (0 to 191)
-f <facility no>	Message facility number (0 to 23)
-l <level no>	Message level number (0 to 7)
-F <facility name>	Message facility name (kernel to local7)
-L <level name>	Message level name (emergency to debug)
-r <process name>	Sends messages in RFC3164 format
-t	Use TCP sockets instead of UDP
-w	Repeat the message every second until a key press
-s	Silently log error messages
-m <message>	The message text
-i	Use standard input for message text (stdin)

All command line switches are optional. Message text must be specified by either the -m or -i switch.

The -i switch allows the input to come from standard input. For example tail -f myfile.txt | Klog -i

The default destination syslog host is *localhost*.

By default, syslog messages are sent with the priority *user.info*. You may change this priority by

- Using the -F and -L flags to modify the facility and level names,
- Using the -f and -l flags to modify the facility and level numbers, or
- Using the -p flag to set the priority using an absolute natural number, up to 191.

Using a –p option will override –f and –l options, which in turn override –F and –L options.

Messages can be sent with a RFC compliant header field. The RFC header is inserted between the message priority value and the message text. The header contains:

- The current date and time,
- The local machine's hostname, and
- The name of the process sending the syslog message (cannot have white space).

Messages sent over TCP/IP will be appended with a Carriage Return character.

As per RFC3164, the total syslog message length (text plus headers) must not exceed 1024 bytes in length.

Error messages are written on screen. Alternatively, the –s argument will write all error messages to *klogerr.log* in the current working directory. If it fails to write to this file, it will not report any errors.

4 Examples

Send a message to the localhost, priority user.info:

```
      klog –m "It's almost lunchtime"
```

Send a syslog message to host 'rotterdam', with priority user.notice:

```
      klog -h rotterdam -F user -L notice –m "The quick brown fox tripped"
```

Send a TCP message to the 'mailhost' Unix server advising of an email:

```
      klog -h mailhost –t -F daemon -L info –m "Mail message sent by Piero"
```

Send a message to machine 'timesvr' every second until a key is pressed on the keyboard:

```
      klog -h timesvr –w –m "Tick tock"
```

Send a message to the syslog daemon on 'sysloghost' running on UDP port 2110. Use RFC headers, with a process name of 'myproc'. Use a pre-calculated priority of 122:

```
      klog -h sysloghost -u 2110 -r myproc -p 122 –m "Local event occurred"
```

Send the output from a DIR command to host 192.168.1.2:

```
      DIR *.* | klog -h 192.168.1.2 -i
```

5 Return codes

The following error codes are returned by Kiwi Logger to the calling application:

Value	**Meaning**
0	No error
10	Invalid UDP port value
20	Invalid hostname
30	Invalid level number
40	Invalid facility number
50	Invalid level name
60	Invalid facility name
70	Invalid priority number
80	Invalid RFC header process name
90	Invalid message text
100	Error opening local socket
110	Error writing to local socket

6 Syslog message reference

Syslog messages are a common method of event logging, and it is a standard feature of Unix computing systems. Messages received by a Syslog Daemon are usually logged to disk. The messages can be split into different files depending on the priority of the message.

Syslog messages usually consist of a priority, a message header, and the message text.

For example:

```
<180>Jan 14 21:56 seibu [process] This is my message
```

The priority is 180
The message header contains: the current date, originating server, and process name.
The message text is the rest of the message.

The BSD syslog format is discussed further in the RFC3164 at:
http://www.ietf.org/rfc/rfc3164.txt

The Syslog Priority Code

The following facilities and levels are recognised by Kiwi Logger. The formula used for priority is:

*Calculated priority = Facility Number * 8 + Level Number*

Facility name	Alt Name	Number
Kernel	Kern	0
User		1
Mail		2
Daemon		3
Auth		4
Syslog		5
Lpr		6
News		7
UUCP		8
Cron		9
System0	Security	10
System1	FTP	11
System2	NTP	12
System3	Logaudit	13
System4	Logalert	14
System5	Clock	15
Local0		16
Local1		17
Local2		18
Local3		19
Local4		20
Local5		21
Local6		22
Local7		23

Level name	Alt name	Number
Emergency	Emerg	0
Alert		1
Critical	Crit	2
Error	Err	3
Warning	Warn	4
Notice		5
Information	Info	6
Debug		7

7 KLOG.DLL available functions

Kiwi Logger is distributed with KLOG.DLL. This DLL provides applications the functions required to send syslog messages. Software developers are encouraged to use this DLL to easily add syslog support directly into their products. The DLL will need to be distributed with the application. Alternatively, you can use the import library to remove the distribution requirement.

```
BOOL WINAPI KLogInitialise();
```

Sets all default values, and erases anything set by other arguments. It MUST be run at least once, and before any other KLOG function.

```
BOOL WINAPI KLogSetPortByService(char *);
```

Sets the destination port for syslog messages using a service name, the default is "syslog", ie port 514. KLOG will create a new connection to the host when the port is set or changed.

```
BOOL WINAPI KLogSetPortByNum(int);
```

Sets the destination port for syslog messages using a number, like –u. KLOG will create a new connection to the host when the port is set or changed.

```
BOOL WINAPI KLogSetHostAddress(char *);
```

Sets the destination hostname or IP address, like –h. KLOG will create a new connection to the host when the hostname is set or changed.

```
BOOL WINAPI KLogSetMessagePriority(int);
```

Sets the message priority, like –p. Setting the priority as a number will override settings from facility numbers, which in turn override settings from facility names.

```
BOOL WINAPI KLogSetMessageLevelNum(int);
```

Sets the message level, like –l.

```
BOOL WINAPI KLogSetMessageFacilityNum(int);
```

Sets the message facility, like –f.

```
BOOL WINAPI KLogSetMessageLevelName(char *);
```

Sets the message level name, like –L.

```
BOOL WINAPI KLogSetMessageFacilityName(char *);
```

Sets the message facility name, like –F.

```
BOOL WINAPI KLogSetProcessName(int, char *);
```

Sets whether or not to use the RFC format (1 or 0). If 1, then the second argument string is the process name, like –r.

```
BOOL WINAPI KLogUseTCP(int);
```

Sets whether to send messages in TCP connections (1) or UDP (0), like –t. KLOG will create a new connection to the host when this is set or changed.

```
BOOL WINAPI KLogUseSilentErrors(int);
```

Sets the silent error option (0 or 1), like –s. Errors are written to klogerr.log in the current directory.

```
BOOL WINAPI KLogSetMessage(char *);
```

Sets the syslog message string.

```
BOOL WINAPI KLogSendMessage(int *);
```

Checks that the message parameters previously set are valid, then attempts to send the message. In the return parameter, it returns 0 on success, or a return code on failure. Use KlogReturnError() to find what happened.

```
BOOL WINAPI KLogReturnError(char *, int);
```

Returns a string message of the current error. The arguments needed are a local string to write the error, and the size of this string.

```
BOOL WINAPI KLogGetUsageHelp(char *, int);
```

Returns generic KLOG usage. Requires a local string to return the details in, and its length.

```
BOOL WINAPI KLogGetUsageExample(char *, int);
```

Returns a generic KLOG usage example. Requires a local string to return the details in, and its length.

```
BOOL WINAPI KLogGetSoftwareInfo(char *, int);
```

Returns copyright details. Requires a local string to return the details in, and its length.

```
BOOL WINAPI KLogGetPriorityNames(char *, int);
```

Returns KLOG priority facilities and levels. Requires a local string to return the details in, and its length.

8 KLOG.DLL C run-time usage

The following C program is a simple example:

- It loads KLOG.DLL at run-time. It assumes it is available in the same directory or in the system root directory
- It gets all functions exported by KLOG.DLL,

- It uses the functions to set parameters, and
- Finally, it sends a message to host 'sysloghost' port 2110, and reports any errors.

```
/*
  Send a syslog message using KLOG.DLL. This is the same as the command
  klog -h sysloghost -u 2110 -r myproc -p 122 –m "Local event occurred"
*/

#include <stdio.h>
#include <windows.h>

//DLL functions
FARPROC KLogInitialise;
FARPROC KLogSetPortByService;
FARPROC KLogSetPortByNum;
FARPROC KLogSetHostAddress;
FARPROC KLogSetMessagePriority;
FARPROC KLogSetMessageLevelNum;
FARPROC KLogSetMessageFacilityNum;
FARPROC KLogSetMessageLevelName;
FARPROC KLogSetMessageFacilityName;
FARPROC KLogSetProcessName;
FARPROC KLogUseTCP;
FARPROC KLogUseSilentErrors;
FARPROC KLogSetMessage;
FARPROC KLogSendMessage;
FARPROC KLogReturnError;
FARPROC KLogGetUsageHelp;
FARPROC KLogGetUsageExample;
FARPROC KLogGetSoftwareInfo;
FARPROC KLogGetPriorityNames;

void getAllFunctions(HMODULE);

int main()
{
  int  UdpPort = 2110;
  char Loghost[] = "sysloghost";
  int  Priority = 122;
  char Processname = "myproc";
  char Msg[] = "Local event occurred";

  int  Kret = 0;      // Returned error code
  char Kerror[100];   // Returned error message
  HMODULE KLib;       // DLL handle

  // Load library, hopefully in current directory.
  if ((KLib = LoadLibrary("klog.dll")) == NULL)
  {
    fprintf(stderr, "Cannot find klog.dll.\nCannot send messages.\n");
    return 1;
  }

  // Get all DLL functions
  getAllFunctions(KLib);

  // Send the message
  KLogInitialise();                         // Do defaults
  KLogSetPortByNum(UdpPort);                // Set destination port
  KLogSetHostAddress(Loghost);              // Set host
  KLogSetMessagePriority(Priority);         // Priority
  KLogSetProcessName(1, Processname);       // RFC process name
  KLogSetMessage(Msg);                      // Set message text

  // Send the message. If FALSE, then check Kret for a return code.
  // Use the DLL to get an error string. If silent errors is enabled,
  // no string is returned in Kerror.
  if (KLogSendMessage(&Kret) == FALSE)
  {
     KLogReturnError(Kerror, sizeof(Kerror));

     if (Kerror[0] != '\0')
        fprintf(stderr, "Error %d: %s\n", Kret, Kerror);
```

```
      return 1;
    }

    return 0;
}

/*
  Get all functions from the DLL.
*/

void getAllFunctions(HMODULE KLib)
{
  KLogInitialise = GetProcAddress(KLib, "KLogInitialise");
  KLogSetPortByService = GetProcAddress(KLib, "KLogSetPortByService");
  KLogSetPortByNum = GetProcAddress(KLib, "KLogSetPortByNum");
  KLogSetHostAddress = GetProcAddress(KLib, "KLogSetHostAddress");
  KLogSetMessagePriority = GetProcAddress(KLib, "KLogSetMessagePriority");
  KLogSetMessageLevelNum = GetProcAddress(KLib, "KLogSetMessageLevelNum");
  KLogSetMessageFacilityNum = GetProcAddress(KLib, "KLogSetMessageFacilityNum");
  KLogSetMessageLevelName = GetProcAddress(KLib, "KLogSetMessageLevelName");
  KLogSetMessageFacilityName = GetProcAddress(KLib, "KLogSetMessageFacilityName");
  KLogSetProcessName = GetProcAddress(KLib, "KLogSetProcessName");
  KLogUseTCP = GetProcAddress(KLib, "KLogUseTCP");
  KLogUseSilentErrors = GetProcAddress(KLib, "KLogUseSilentErrors");
  KLogSetMessage = GetProcAddress(KLib, "KLogSetMessage");
  KLogSendMessage = GetProcAddress(KLib, "KLogSendMessage");
  KLogReturnError = GetProcAddress(KLib, "KLogReturnError");
  KLogGetUsageHelp = GetProcAddress(KLib, "KLogGetUsageHelp");
  KLogGetUsageExample = GetProcAddress(KLib, "KLogGetUsageExample");
  KLogGetSoftwareInfo = GetProcAddress(KLib, "KLogGetSoftwareInfo");
  KLogGetPriorityNames = GetProcAddress(KLib, "KLogGetPriorityNames");
}
```

9 KLOG.LIB C import library

Developers can use the KLOG.LIB import library to implicitly access Kiwi Logger functions. The system will load KLOG.DLL, and its functions automatically at load time. This method is suitable for C and C++ projects. KLOG.DLL will need to be distributed with the application.

To include the import library in your project:
- Make KLOGLIB.H and KLOG.LIB available to your project,
- Include the header file in your code, such as with `#include "kloglib.h"`, and
- Include KLOG.LIB in the link step of your project. In Visual C++, refer to the *Project Settings* dialogue by pressing Alt+F7, and the *Link* tab.

10 KLOG.DLL C++ class sample

The following code is a sample C++ class which applies Kiwi Logger functions. It is compiled using the KLOG.LIB import library.

It provides the following methods:
- A constructor which sets the syslog host, port, transport type, and process header,
- A method to set the priority using facility names,
- A method to set the message to send,
- A method to send the message, and
- A method to return any resulting errors.

Klogobj.h class definition:

```
/*
  Klogobj.h

  Kiwi Logger v1.3
  Sample C++ Logger class.
  Links implicitly with KLOG.LIB.
  Accesses KLOG.DLL at load-time.
*/

#include "kloglib.h"

class KLog
{
private:
  int  Kret;             // Returned error code
  char Kerror[100];      // Returned error string

public:

  // Constructor. Initialise storage. Initialise the DLL.
  // Set the syslog host, port, IP protocol, and RFC header process name.
  KLog(char *Sysloghost, char *Service, int Protocol, char *Processname)
  {
    Kret = 0;
    memset(Kerror, 0, sizeof(Kerror));
    KLogInitialise();
    KLogSetHostAddress(Sysloghost);
    KLogSetPortByService(Service);
    KLogUseTCP(Protocol);
    KLogSetProcessName(1, Processname);
  }

  // Deconstructor. Do nothing.
  ~KLog()
  {
  }

  // Set the priority using facility names
  void SetPriority(char *FacilityName, char *LevelName)
  {
    KLogSetMessageFacilityName(FacilityName);
    KLogSetMessageLevelName(LevelName);
  }

  // Set the message
  void SetMessage(char *Msg)
  {
    KLogSetMessage(Msg);
  }

  // Send the message. Return FALSE if an error occurred.
  BOOL SendMessage()
  {
    if (KLogSendMessage(&Kret) == FALSE)
    {
       KLogReturnError(Kerror, sizeof(Kerror));
       return FALSE;
    }
    else
       Kerror[0] = '\0';

    return TRUE;
  }

  // Return the last error code and message
  void ReturnError(int *Rret, char *Rerror, int Rlen)
  {
    *Rret = Kret;
    strncpy(Rerror, Kerror, Rlen);
  }
};
```

Example message code:

```
// Klog C++ object usage example

#include <iostream.h>
#include "klogobj.h"

int main(int argc, char* argv[])
{
  char Loghost[] = "sunloghost";
  char Myprocess[] = "oracle_watch";
  char Message[] = "Oracle listener is not active on port 1521.";

  int  Retcode;              // Return code and error
  char Reterr[100];

  // Create a new KLog Logger object.
  // Send message to sunloghost, using UDP (0) on port "syslog".
  KLog MyKLog(Loghost, "syslog", 0, Myprocess);

  MyKLog.SetPriority("daemon", "error");
  MyKLog.SetMessage(Message);

  if (MyKLog.SendMessage() == FALSE)
  {
    MyKLog.ReturnError(&Retcode, Reterr, sizeof(Reterr));
    cout << "Error " << Retcode << " occurred: " << Reterr << endl;
  }

  return 0;
}
```

11 KLOG.DLL access from Visual Basic

An alternative to the KLOGCTL ActiveX control in the next section, is to access syslog functionality directly from KLOG.DLL. The Visual Basic sample shows the function declarations. Code is provided to send a simple message, then check for an error.

```
'
' Kiwi Logger DLL declarations
'
Public Declare Function KLogInitialise Lib "klog" () As Boolean
Public Declare Function KLogSetPortByService Lib "klog" (ByVal UDPPortOut As String) As
Boolean
Public Declare Function KLogSetPortByNum Lib "klog" (ByVal UDPPortOut As Long) As Boolean
Public Declare Function KLogSetHostAddress Lib "klog" (ByVal SysHostOut As String) As Boolean
Public Declare Function KLogSetMessagePriority Lib "klog" (ByVal PriorityOut As Long) As
Boolean
Public Declare Function KLogSetMessageLevelNum Lib "klog" (ByVal LevNoOut As Long) As Boolean
Public Declare Function KLogSetMessageFacilityNum Lib "klog" (ByVal FacNoOut As Long) As
Boolean
Public Declare Function KLogSetMessageLevelName Lib "klog" (ByVal LevNameOut As String) As
Boolean
Public Declare Function KLogSetMessageFacilityName Lib "klog" (ByVal FacNameOut As String) As
Boolean
Public Declare Function KLogSetProcessName Lib "klog" (ByVal UseRFCOut As Long, ByVal
PIDNameOut As String) As Boolean
Public Declare Function KLogUseTCP Lib "klog" (ByVal UseTCPOut As Long) As Boolean
Public Declare Function KLogUseSilentErrors Lib "klog" (ByVal SilentErrOut As Long) As Boolean
Public Declare Function KLogSetMessage Lib "klog" (ByVal MessageOut As String) As Boolean
Public Declare Function KLogSendMessage Lib "klog" (Rcode As Long) As Boolean
Public Declare Function KLogReturnError Lib "klog" (ByVal ErrorIn As Long, ByVal ErrLen As
Long) As Boolean
Public Declare Function KLogGetUsageHelp Lib "klog" (ByVal HelpIn As Long, ByVal HelpInLen As
Long) As Boolean
Public Declare Function KLogGetUsageExample Lib "klog" (ByVal HelpExIn As Long, ByVal
HelpExInLen As Long) As Boolean
Public Declare Function KLogGetSoftwareInfo Lib "klog" (ByVal SwInfoIn As Long, ByVal
SWInfoInLen As Long) As Boolean
```

```
Public Declare Function KLogGetPriorityNames Lib "klog" (ByVal PriNameIn As Long, ByVal
PriNameLen As Long) As Boolean

'---------------------------------------------------------------------

' Uses the KLog DLL functions to set the syslog options, as per
' klog -h sysloghost -u 2110 -r myproc -p 122 –m "Local event occurred"
' The DLL handles all error checking, except for VB empty strings or values.

Private Sub SendSimpleMessage()

Dim Rcode As Long

KLogInitialise
KLogSetHostAddress "Sysloghost"
KLogSetPortByNum 2110
KLogSetMessagePriority 122
KLogSetProcessName 1, "myproc"
KLogUseSilentErrors 0
KLogSetMessage "Local event occurred"

KLogSendMessage Rcode

' Exit on success
If Rcode = 0 Then
  Exit Sub
End If

' Returns the last error from KLOG.DLL.
' Pass a pointer to a byte array.
' Convert the result into a string.

Dim ByteArray(201) As Byte
Dim BytePtr As Long
Dim ReturnString As String

BytePtr = VarPtr(ByteArray(0))
KLogReturnError BytePtr, CLng(UBound(ByteArray) - 1)

' If null, then no string returned, since silent errors is enabled
If Not ByteArray(0) = 0 Then
  ReturnString = StrConv(ByteArray, vbUnicode)
  MsgBox ReturnString, vbInformation, "KLOG.DLL error " & Rcode
End If

End Sub
```

12 KLOG_COM.DLL - VB/ASP/VBScript

The *Kiwi Logger COM object (KLOG_COM.dll)* is designed for use with VB, ASP, Windows Scripting Host (ie. VBScript) and any other language that supports COM.

Name: Klog_COM.DLL

Purpose: To provide basic UDP syslog message sending capabilities via a COM DLL with minimum dependencies

Works with: VB, VBScript, ASP etc.

Dependencies: MSWINSCK.OCX, MSVBVM60.DLL

Properties:

RemoteAddress(strRemoteAddress As String)

RemotePort(lngRemotePort As Long)

FacilityName(strFacilityName As String)

FacilityNumber(lngFacilityNumber As Long)

LevelName(strLevelName As String)

LevelNumber(lngLevelNumber As Long)

PriorityNumber(lngPriorityNumber As Long)

MessageText(strMessageText As String)

UseRFC(blnUseRFC As Boolean)

UseFile(blnUseFile As Boolean)

FileName(strFileName As String)

UseUTF8(blnUseUTF8 As Boolean)

Error_Data(strError_Data As String)

Methods:

```
Function SendMessage(Optional sRAddress As String, Optional lRPort As Long, _
                Optional vFac As Variant, Optional vLev As Variant, _
                Optional sMText As String, Optional bRFC As Boolean, _
                Optional bFile As Boolean, Optional sFile As String, _
                Optional bUFT8 As Boolean) As Boolean
```

Function UTF8_Encode(strUnicode As String) As Byte()

Function FileWrite(fName As String, fData As String) As Boolean

<u>VBScript Example:</u>

```
   Option Explicit

   Dim oSysSender
   ' Create the Syslog Sender object
   Set oSysSender = CreateObject("Klog_COM.SyslogSender")

   ' Set the properties
   oSysSender.FacilityName = "local7"     ' Local7 Facility
   oSysSender.LevelName = "info"          ' Info level
   oSysSender.RemoteAddress = "127.0.0.1" ' Send to local host
   oSysSender.RemotePort = "514"          ' Use the standard UDP Syslog
port
   oSysSender.UseFile = False             ' Send via UDP
   oSysSender.UseRFC = False              ' Don't use the RFC3164 format
   oSysSender.UseUTF8 = True              ' Encode the data as UTF8.
   oSysSender.MessageText = "This is a test message send from KlogX.DLL"

   ' Now send the message
   Dim RetVal
   RetVal = oSysSender.SendMessage()

   ' Notify if message was sent successfully
   If RetVal = True then
       MsgBox "Message sent successfully"
   Else
       MsgBox "Message was not sent successfully. Error: " &
oSysSender.Error_Data
   End if

   ' Destroy the object
   Set oSysSender = Nothing
```

ASP Example:

```
<HTML>
<HEAD><TITLE>Syslog Sender Test</TITLE></HEAD>
<BODY>
<%

   Dim oSysSender
   ' Create the Syslog Sender object
   Set oSysSender = CreateObject("Klog_COM.SyslogSender")

   ' Set the properties
   oSysSender.FacilityName = "local7"     ' Local7 Facility
   oSysSender.LevelName = "info"          ' Info level
   oSysSender.RemoteAddress = "127.0.0.1" ' Send to local host
   oSysSender.RemotePort = "514"          ' Use the standard UDP Syslog
port
   oSysSender.UseFile = False             ' Send via UDP
   oSysSender.UseRFC = False              ' Don't use the RFC3164 format
   oSysSender.UseUTF8 = True              ' Encode the data as UTF8.

   oSysSender.MessageText = "This is a test message send from KlogX.DLL"

   ' Now send the message
   Dim RetVal
   RetVal = oSysSender.SendMessage()

   ' Notify if message was sent successfully
   If RetVal = True then
       Response.write "Message sent successfully"
   Else
       Response.write "Message was not sent successfully. Error: " &
oSysSender.Error_Data
   End if

   ' Destroy the object
   Set oSysSender = Nothing

%>
</BODY>
</HTML>
```

VB6 Example:

```
   ' Create the Syslog Sender object
   Dim oSysSender As New KLog_COM.SyslogSender

   ' Set the properties
   oSysSender.FacilityName = "local7"     ' Local7 Facility
   oSysSender.LevelName = "info"          ' Info level
   oSysSender.RemoteAddress = "127.0.0.1" ' Send to local host
   oSysSender.RemotePort = "514"          ' Use the standard UDP Syslog
port
   oSysSender.UseFile = False             ' Send via UDP
   oSysSender.UseRFC = False              ' Don't use the RFC3164 format
   oSysSender.UseUTF8 = True              ' Encode the data as UTF8.

   oSysSender.MessageText = "This is a test message send from KlogX.DLL"

   ' Now send the message
   Dim RetVal
   RetVal = oSysSender.SendMessage()

   ' Notify if message was sent successfully
   If RetVal = True Then
       MsgBox "Message sent successfully"
   Else
       MsgBox "Message was not sent successfully. Error: " &
oSysSender.Error_Data
   End If

   ' Destroy the object
   Set oSysSender = Nothing
```

13 KLOG_NET.DLL - VB.net/ASP.net/C#

The *Kiwi Logger .Net DLL (KLOG_NET.dll) is a multi-threaded .net DLL that has been* designed specifically for use with VB.net, ASP.net or other .net languages (eg. C#).

Requirements: .Net Framework

Properties:

RemoteAddress*(strRemoteAddress As String)*

RemotePort*(lngRemotePort As Long)*

FacilityName*(strFacilityName As String)*

FacilityNumber*(lngFacilityNumber As Long)*

LevelName*(strLevelName As String)*

LevelNumber*(lngLevelNumber As Long)*

PriorityNumber*(lngPriorityNumber As Long)*

MessageText*(strMessageText As String)*

UseRFC*(blnUseRFC As Boolean)*

UseUTF8*(blnUseUTF8 As Boolean)*

ErrorData*(strError_Data As String)*

Methods:

InitSocket*(useTCP as Boolean) as Boolean*

- Initializes the socket. Used to set the kind of data the socket will be sending.
- Called with one parameter: UseTCP = true/false, False=UDP protocol, True=TCP protocol
- InitSocket() Returns TRUE if the Socket type was set successful.

Connect*() as Boolean*

- Performs an asynchronous connection on an initialized socket.
- The callback delegate will fire the Connect() event when connected.
- Returns TRUE if Asynchronous connect has begun successfully.

```
SendMessage( Optional sRAddress As String, Optional lRPort As Long, _
             Optional vFac As Variant, Optional vLev As Variant, _
             Optional sMText As String, Optional bRFC As Boolean, _
             Optional bUFT8 As Boolean   ) as Boolean
```

- Performs an asynchronous send.
- The callback delegate will fire the SendComplete() event when sent.

Events:

Klog_Net.dll raises separate events which signal the completion of the callback delegate or worker thread.

The events are:

Connected*()*
- Used for TCP Protocol-Types, not necessary for UDP (A connection-less protocol)
- Indicates that the Connect() method has been successful.

SendComplete*(bytesSent as Long)*
- Signals the completion of a send process.

Exception*(ex as Exception)*
- Exposes any exception generated by callback delegates or in the DLL itself.

Examples:

Samples can be found in **../KLOGDLL_Net/Samples.** There is one sample solution for ASP.net (KlogWeb) and one for VB.net (KLogWin).
KLOG_NET.dll can be found linked in the /bin directory of each sample, or unlinked in the ../KLOGDLL_NET directory.

The basic structure of a .net application that utilizes KLog_Net.dll is as follows:

```
' UDP SYSLOG SENDER
' ===============
' Ensure that KLOG_NET.dll is added as a reference

' Create instance of KLOG_NET.SyslogSender
Dim SyslogSender as New KLOG_NET.SyslogSender

' Set the type of socket and then initialize that socket
SyslogSender.InitSocket(false)                  ' USE UDP!

' Set the Remote Host
SyslogSender.RemoteAddress = "127.0.0.1"        ' (Localhost)
SyslogSender.RemotePort = "514"                 ' Standard Syslog UDP port

' Set the message options
SyslogSender.FacilityNumber = 1
SyslogSender.LevelNumber = 1
SyslogSender.UseUTF8 = True               ' Use UTF8 Encoding

' Set the message content
SyslogSender.MessageText = "Hello world"

' Send the message
SyslogSender.SendMessage()

' Shutdown and close the Socket
SyslogSender.Close()
SyslogSender = Nothing
```

14 License Agreement

Terms and Conditions of use:
=============================
The Kiwi Logger (KLOG) components and command line tools are released as freeware and can be used royalty-free subject to the conditions below.

By installing and/or using Kiwi Logger (KLOG) - (SOFTWARE PRODUCT), you agree NOT to:
==
(a) Decompile, reverse engineer, disassemble or modify the SOFTWARE PRODUCT or the documentation in whole or in part.
(b) Remove any copyright or other Kiwi Enterprises proprietary notices.

Redistributable Components:
============================
The following files are designated as "REDISTRIBUTABLE COMPONENTS" and are subject to the Distribution Requirements below.

KLOG.exe - Windows console based syslog sender

KLOGWin.exe - Windows based syslog sender
KLOG.DLL - Windows DLL component for use with C++
KLOG_COM.DLL - Windows COM based DLL for use with VB/ASP/Scripting
KLOG_NET.DLL - Windows .Net based DLL for use with .Net

Distribution Requirements:
==========================
You are authorized to redistribute the REDISTRIBUTABLE COMPONENTS as listed above, only if you:

(a) distribute them in conjunction with and as part of your own software application that adds primary and significant functionality to the REDISTRIBUTABLE COMPONENTS;
(b) do not permit further redistribution of the REDISTRIBUTABLE COMPONENTS by your end-user customers;
(c) do not use the Kiwi Enterprises name, logo, or trademarks to market your software application;
(d) include a valid copyright notice on your software application; and
(e) agree to indemnify, hold harmless, and defend Kiwi Enterprises from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of your software application.

Kiwi Enterprises reserves all rights not expressly granted.

The license in this section to distribute REDISTRIBUTABLE COMPONENTS is royalty-free, provided that you do not make any modifications to any of the REDISTRIBUTABLE COMPONENTS. Please contact Kiwi Enterprises for the applicable royalties due and other licensing terms for all other uses and/or distribution of the REDISTRIBUTABLE COMPONENTS.

Termination of licence:
=======================
Without prejudice to any other rights, Kiwi Enterprises may terminate this licence agreement if you fail to comply with the terms and conditions contained within this licence agreement. In such event, you must destroy all copies of the SOFTWARE PRODUCT and all of its component parts, including any registration keys.

Ownership:
==========
The SOFTWARE PRODUCT is copyrighted proprietary material of Kiwi Enterprises and may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without Kiwi Enterprises prior written permission.

To obtain permission, please contact Kiwi Enterprises via e-mail at:
support@kiwisyslog.com

Software product licence:
=========================
The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

Disclaimer:
===========
The SOFTWARE PRODUCT is provided "AS-IS" without warranty of any kind either express or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose or non-infringement. Some jurisdictions do not allow or otherwise govern the scope of exclusions of implied warranties, so the above exclusions my not apply in full.

The SOFTWARE PRODUCT may contain technical inaccuracies or typographical errors, so changes and/or updates may be affected without notice.

Kiwi Enterprises may also make improvements and/or other changes to the SOFTWARE PRODUCT at any time without notice.

Not for use in high risk activities:
====================================
This SOFTWARE PRODUCT is not fault-tolerant and is not designed, manufactured, or intended for use, or resale, in hazardous environments requiring fail-safe performance. Such environments and systems include the operation of nuclear facilities, aircraft navigation, aircraft communication systems, air traffic control, direct life support machines, weapons systems, or any environment or system in which the failure of this SOFTWARE PRODUCT could lead directly, or indirectly, to death, personal injury, or severe physical or environmental damage.

Kiwi Enterprises specifically disclaims any express or implied warranty of fitness for use of this SOFTWARE PRODUCT in High Risk Activities.

No liability for consequential damages:
=======================================
To the maximum extent permitted by applicable law, in no event shall Kiwi Enterprises be liable for any damages whatsoever (including, without limitation, damages for loss of business profit, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of, or inability to use, this Kiwi Enterprises product, even if Kiwi Enterprises has been advised of the possibility of such damages.