BEUtilities

This document discusses the operation of the BEUtilities.dll.

COMPONENT OVERVIEW

The BEUtilities.dll file is a collection of general purpose functions that can be used by any Win32 application. The collection of functions include:

1) NT Event Logging
2) File Transfer

NT EVENT LOGGING

Typically, NT Event logging capabilities are built into an application using the ReportEvent API. However, this is not without some complication. Various files, created with mc.exe (found on the Microsoft SDK disk), must be integrated into a project and compiled with the application.

The BEUtilities.dll bypasses these complicated procedures and allows any application that can call a DLL to write to the Event Viewer directly. EventLogWrite supports the error, warning, and information event types.

Usage

Before the application using EventLogWrite can write to the NT Event Log, it must be registered. This is accomplished by calling the exported function “EventLogRegisterApp”, ***only once***, and passing in your application name as a string parameter. For example, in Visual Basic, the code would look something like this:

```
Option Explicit
Private Declare Function EventLogRegisterApp Lib " BEUtilities.dll" (ByVal szAppTitle As String) As Long

Private Sub MyFunction()
Dim lRetVal As Long

lRetVal = EventLogRegisterApp("MyApp")
if lRetVal <> 0 then Call MsgBox(“Error”) ‘zero (0) means success

End Sub
```

After this function has been called, an application is ready to write to the NT Event Viewer. Repeated calls to this function won’t hurt anything, but it is not recommended due to the risk involved in continually updating the registry unnecessarily.

To write a message to the NT Event Log, the “EventLogWriteWrite” exported function must be called. “EventLogWriteWrite” accepts 3 parameters. The first is a WORD (long) indicating the type of message. Valid types are:

```
EVENTLOG_ERROR_TYPE = 0x0001
EVENTLOG_WARNING_TYPE = 0x0002
EVENTLOG_INFORMATION_TYPE = 0x0004
```

Any other values will cause the function to return a non-zero value (non-success). The 2nd parameter is a LPSTR (string) containing the name used to register the application in EventLogRegisterApp. The application names must match. The 3rd parameter is a LPSTR (string) containing the message you would like to have displayed in the NT event viewer. A example of this usage is as follows:

```
Option Explicit
Private Declare Function EventLogWriteWrite Lib "BEUtilities.dll" (ByVal lType As Long, ByVal szAppTitle As String, ByVal szMessage As String) As Long

Private Sub MyFunction()
Dim lRetVal As Long
Const EVENTLOG_ERROR_TYPE = 1
Const EVENTLOG_WARNING_TYPE = 2
Const EVENTLOG_INFORMATION_TYPE = 4

'Write an information type message to the NT Event Viewer Log. Note 'that the application name must match the name that was registered.
lRetVal = EventLogWriteWrite (EVENTLOG_INFORMATION_TYPE, "MyApp", "This is my message")

if lRetVal <> 0 then Call MsgBox("Error") 'zero (0) means success

End Sub
```

FILE TRANSFER

The BEUtilities .dll has functionality to transfer files with "watchdog" and retry capabilities. There is a function called FileTransferVB that can be called from a Visual Basic application (or any applications passing a BSTR) and another called FileTransferC for Microsoft C++ applications (or any application that can use a Cstring). Each function accepts the same parameters.

Parameters

1st parameter

A **string** containing the source path and file name. This path and file name will be checked to make sure it exists, or the function will return an error code.

2nd parameter

A **string** containing the destination path and file name.

3rd parameter

A **string** value not used at the present time.

4th parameter

A **string** value not used at the present time.

5th parameter

A **string** value not used at the present time.

6th parameter

A **long** value containing the retry value. If the file transfer function fails it will retry the amount of times specified in this parameter before returning an error code. ***The highest this value can be set to is 10.***

7th parameter

A **long** value containing the timeout value. This is the maximum amount of time, in seconds, that the application will wait for a file transfer to occur. Care must be taken to ensure this value is set high enough to allow the transfer of large files so an error isn't detected too early.

8th parameter

A **long** value containing the file transfer mode. The available mode are:

1 – File copy. The source file will be copied to the destination.
2 – File move. The source file will be moved to the destination. Attempting to move a file to the same location or moving a file to another location with a different name will cause an error code to be returned.
3 – File copy and delete. The source file will be copied to the destination path and deleted upon completion. With this method, a file can moved to another location as another name although it is not required.

9th parameter

A **long** value specifying whether or not the function should force a file to be overwritten during copy and move operations. If this value is 0, the function will fail if the destination file already exists. If this value is 1, the function will overwrite the destination file.

Please note that in order to delete or overwrite a read-only file, the attributes will be reset to remove the read-only flag. Rights on the target machine must exist allowing the changing of attributes for this functionality to work.

10th parameter

A **long** value specifying how many seconds the function should wait before retrying in the event of an error.

Return Codes

0 – No errors. Function successful
1 – Invalid source file name.
2 – Invalid destination file name or path.
3 – Bad transfer mode. An invalid transfer mode set.
4 – Retry limit set to high. Maximum of 10 retries.
5 – Source file does not exist.
6 – File transfer error. An unknown error occurred attempting to transfer a file.

If a file transfer error occurs, an NT event log entry will be generated under “BEUtilities” in the Applications log and can be viewed with the NT event Viewer. This may be useful in diagnosing file transfer errors.

Usage Example

```
Private Declare Function FileTransferVB Lib “BEUtilities.dll" _
                                                  (ByVal szSourceFile As String, _
                                                  ByVal szDestinationPath As String, _
                                                  ByVal szParam3 As String, _
                                                  ByVal szParam4 As String, _
                                                  ByVal szParam5 As String, _
                                                  ByVal lRetryVal As Long, _
                                                  ByVal lTimeOutVal As Long, _
                                                  ByVal lTransferMode As Long, _
                                                  ByVal lForceOverwrite As Long, _
                                                  ByVal lRetryDwell As Long) As Long


Private Const MAX _RETRIES = 10
Private Const MODE_FILE_COPY = 1
Private Const MODE_FILE_ MOVE = 2
Private Const MODE_FILE_ COPYANDDELETE = 3

Private Const ERROR_NONE = 0
Private Const ERROR_BAD_SOURCEFILE_NAME = 1
Private Const ERROR_BAD_DESTINATIONFILE_NAME = 2
Private Const ERROR_BAD_TRANSFER_MODE = 3
Private Const ERROR_RETRY_LIMIT_TOO_HIGH = 4
Private Const ERROR_SOURCE_FILE_NO_EXIST = 5
Private Const ERROR_FILE_TRANSFER = 6

Private Const DO_NOT_FORCE_OVERWRITE = 0
Private Const FORCE_OVERWRITE = 1

Private Sub TransferMyFile()
Dim n_strSourceFile as String
Dim n_strDestinationFile as String
Dim n_lReturnValue as Long

n_strSourceFile = “C:\myDir\TextFile.txt”
n_strDestinationFile = “C:\NewDir\TextFile.txt”

n_lReturnValue = FileTransferVB(n_strSourceFile, _
                        n_strDestinationFile, _
                        vbNull, _
                        vbNull, _
                        vbNull, _
                        3, _ ‘Retry 3 times
                        300, _ ‘wait 5 minutes
                        MODE_FILE_COPY, _ ‘Copy file
                        FORCE_OVERWRITE, _ ‘Overwrite if it exists
                        5) ‘Wait 5 seconds before retrying

If n_lReturnValue <> 0 Then Call MsgBox(“Error”)

End Sub
```