Reading and Writing to Windows Event Log

-
StPendl

Read Event Log | Write Event Log

Read Event Log

```
struct EVENTLOGRECORD, _
        Length              as uLong, _
        Reserved            as uLong, _
        RecordNumber        as uLong, _
        TimeGenerated       as uLong, _
        TimeWritten         as uLong, _
        EventID             as uLong, _
        EventType           as word, _
        NumStrings          as word, _
        EventCategory       as word, _
        ReservedFlags       as word, _
        ClosingRecordNumber as uLong, _
        StringOffset        as uLong, _
        UserSidLength       as uLong, _
        UserSidOffset       as uLong, _
        DataLength          as uLong, _
        DataOffset          as uLong

    Open "advapi32.dll" for dll as #advapi32

    lpSourceName$ = "Application"; chr$(0)

    calldll #advapi32, "OpenEventLogA", _
        lpUNCServerName as ulong, _
        lpSourceName$   as ptr, _
        hEventLog       as ulong

    print
    print "Open Event Log Handle: "; hEventLog

    if hEventLog = 0 then call DisplayError

    struct OldestRecord, value as ulong

    calldll #advapi32, "GetOldestEventLogRecord", _
        hEventLog    As uLong, _
        OldestRecord as struct, _
        result       as long
```

```
    print
    print "Oldest Event Log result: "; result
    print "Oldest Event Log Number: "; OldestRecord.value.struct

    if result = 0 then call DisplayError

    struct NumberOfRecords, value as ulong

    calldll #advapi32, "GetNumberOfEventLogRecords", _
        hEventLog       As uLong, _
        NumberOfRecords as struct, _
        result          as long

    print
    print "Number of Event Log Records result: "; result
    print "Number of Event Logs: "; NumberOfRecords.value.struct

    if result = 0 then call DisplayError

    Struct pnBytesRead, value As uLong
    Struct pnMinNumberOfBytesNeeded, value As uLong

    dwReadFlags = _EVENTLOG_SEEK_READ or _EVENTLOG_FORWARDS_READ
    dwRecordOffset = OldestRecord.value.struct +
NumberOfRecords.value.struct - 1
    nNumberOfBytesToRead = hexdec("7ffff")
    lpBuffer$ = space$(nNumberOfBytesToRead); chr$(0)

    calldll #advapi32, "ReadEventLogA", _
        hEventLog               As uLong, _
        dwReadFlags             As uLong, _
        dwRecordOffset          As uLong, _
        lpBuffer$               As ptr , _
        nNumberOfBytesToRead    As uLong, _
        pnBytesRead             As Struct , _
        pnMinNumberOfBytesNeeded As struct , _
        result                  As long

    'print something i can check
    print
    print "Results: "
    print
pnMinNumberOfBytesNeeded.value.struct, pnBytesRead.value.struct
    print "Buffer: "
    print left$(lpBuffer$, pnBytesRead.value.struct)
```

```
    print
    print "Read Event Log result: "; result

    if result = 0 then call DisplayError

    calldll #advapi32, "CloseEventLog", _
        hEventLog as ulong, _
        result    as long

    print
    print "Close Event Log result: "; result

    if result = 0 then call DisplayError

    close #advapi32
    end

sub DisplayError
    calldll #kernel32, "GetLastError", _
        ErrorCode as ulong

    dwFlags = _FORMAT_MESSAGE_FROM_SYSTEM
    nSize = 1024
    lpBuffer$ = space$(nSize); chr$(0)
    dwMessageID = ErrorCode

    calldll #kernel32, "FormatMessageA", _
        dwFlags      as ulong, _
        lpSource     as ulong, _
        dwMessageID  as ulong, _
        dwLanguageID as ulong, _
        lpBuffer$    as ptr, _
        nSize        as ulong, _
        Arguments    as ulong, _
        result       as ulong

    print "Error "; ErrorCode; ": "; left$(lpBuffer$, result)
end sub
```

Write Event Log

```
    open "advapi32.dll" for dll as #advapi32
```

```
struct lpStrings, string$ as ptr

lpSourceName$ = "Application"; chr$(0)

wType = _EVENTLOG_INFORMATION_TYPE
'    dwEventID = 8194
'    wCategory = 5
wNumStrings = 1
lpStrings.string$.struct = "LB Event Log Test"; chr$(0)

calldll #advapi32, "RegisterEventSourceA", _
    lpUNCServerName as ulong, _   'local computer if 0
    lpSourceName$   as ptr, _     'source eg. application name
    handle          as ulong      'handle for ReportEvent

print
print "Register Event Source Handle: "; handle

if handle = 0 then call DisplayError

calldll #advapi32, "ReportEventA", _
    handle      as ulong, _  'event log handle
    wType       as word, _   'event type
    wCategory   as word, _   'category zero
    dwEventID   as ulong, _  'event identifier
    lpUserSID   as ulong, _  'no user security identifier
    wNumStrings as word, _   'one substitution string
    dwDataSize  as ulong, _  'no data
    lpStrings   as struct, _ 'address of string array
    lpRawData   as ulong, _  'address of data
    result      as long

print
print "Report Event Result: "; result

if result = 0 then call DisplayError

calldll #advapi32, "DeregisterEventSource", _
    handle as ulong, _
    result as long

print
print "Deregister Event Source Result: "; result

if result = 0 then call DisplayError
```

```
    print
    print "Finished ..."

    close #advapi32
    end

sub DisplayError
    calldll #kernel32, "GetLastError", _
        ErrorCode as ulong

    dwFlags = _FORMAT_MESSAGE_FROM_SYSTEM
    nSize = 1024
    lpBuffer$ = space$(nSize); chr$(0)
    dwMessageID = ErrorCode

    calldll #kernel32, "FormatMessageA", _
        dwFlags      as ulong, _
        lpSource     as ulong, _
        dwMessageID  as ulong, _
        dwLanguageID as ulong, _
        lpBuffer$    as ptr, _
        nSize        as ulong, _
        Arguments    as ulong, _
        result       as ulong

    print "Error "; ErrorCode; ": "; left$(lpBuffer$, result)
end sub
```

Read Event Log | Write Event Log