Advanced Exception Information Format (AXIF)
draft-luis140219-axif-02

Abstract

This document defines the Advanced Exception Information Format (AXIF), an open and extensible format for providing information about errors and exceptions that allows transmission of information about multiple exceptions and the relations between them.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 18, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .............................................. 2
2. The format .............................................. 2
3. Values for child OIDs of 1.3.6.1.4.1.37476.9000.46.2.19 .... 2
   3.1. NTSTATUS ........................................... 2
   3.2. Level .............................................. 3
1. Introduction

This document specifies the Advanced Exception Information Format (also known as AXIF), a format defined using ASN.1 [X.680].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Microsoft and Windows are registered trademarks of Microsoft Corporation, and Win32 may be a trademark or registered trademark of Microsoft Corporation in some jurisdictions.

The OID 1.3.6.1.4.1.37476.9000.46.2.19 is allocated to this format.

2. The format

The format is the application of the Basic Encoding Rules [X.690] to instances of the AXIF type in the ASN.1 module in Appendix A. As a consequence of the choice of encoding rules, AXIF is a binary file format.

Each ExceptionInformation in an AXIF file/transmission is a list of data structures that contain a property OID, the referencesOthers boolean (Section 4), and a value for the property identified by the OID. Rules for values of properties with child OIDs of 1.3.6.1.4.1.37476.9000.46.2.19 are defined in Section 3.

3. Values for child OIDs of 1.3.6.1.4.1.37476.9000.46.2.19

Unless explicitly stated otherwise for a particular OID, for OIDs in this section that are AXIF properties, the referencesOthers boolean for instances of properties identified by the OID MUST be false.
### 3.1. NSTATUS

This type is used in an ExceptionInformation to indicate the NSTATUS value (Section 2.3.1 of [MS-ERREF]) of the exception corresponding to the ExceptionInformation.

The OID for specifying a value of this type is 1.3.6.1.4.1.37476.9000.46.2.19.1.

### 3.2. Level

This type is used in an ExceptionInformation to indicate the error level of the exception: if it was a direct problem, other problem that involved clients or just other, internal problem. A value of this type is of the ASN.1 type Level, defined in Appendix A.

The OID for specifying a value of this type is 1.3.6.1.4.1.37476.9000.46.2.19.2.

### 3.3. Basic Information

This type is used in an ExceptionInformation to indicate information about the system in which the exception corresponding to the ExceptionInformation occurred. A value of this type is of the ASN.1 type BasicInformation, defined in Appendix A.

The OID for specifying a value of this type is 1.3.6.1.4.1.37476.9000.46.2.19.3.

### 3.4. Reserved

OID 1.3.6.1.4.1.37476.900.46.2.19.4 is reserved for identifying the ASN.1 module in Appendix A.

This section is just here for consistency in the numbering.

### 3.5. x86 Context

This type is used in an ExceptionInformation to indicate the context in which an exception occurred in a x86 system. A value of this type is encoded in ASN.1 as an OCTET STRING with exactly 204, 716 or 720 octets, using the CONTEXT C structure as defined in WinNT.h when targeting x86, being all fields in little-endian. If the system is not x86, or the process that generated the exception is not a x86 32-bit process in a 64-bit operating system, this type MUST NOT be used. With exactly 204 octets, the ExtendedRegisters field is removed from the structure and all attempts to extract data from it MUST fail. The ASN.1 type for encoding values of this type is called X86Context in the ASN.1 module in Appendix A.

The structure’s definition can be obtained at [X86CTX].
The OID for specifying a value of this type is 1.3.6.1.4.1.37476.9000.46.2.19.5.

3.6. Win32 Exception Record

This type is used in an ExceptionInformation to indicate an exception record describing the exception. A value of this type is of the ASN.1 type ExceptionRecord, defined in Appendix A. This ASN.1 type is an 80-octet OCTET STRING representing an EXCEPTION_RECORD [EXCREC] C structure, being all fields in little-endian, and is called Win32ExceptionRecord in the ASN.1 module of Appendix A.

The OID for specifying a value of this type is 1.3.6.1.4.1.37476.9000.46.2.19.6.

3.7. Memory Map

This type is used in an ExceptionInformation to indicate a total or partial memory map of the process in which the exception occurred. A value of this type is of ASN.1 type MemoryMap, defined in Appendix A.

The protection value is an integer working as a bitfield, for which the following flags are defined [MEMINFO] [MEMPROTC]:

* 0x00000001 - PAGE_NOACCESS
* 0x00000002 - PAGE_READONLY
* 0x00000004 - PAGE_READWRITE
* 0x00000008 - PAGE_WRITECOPY
* 0x00000010 - PAGE_EXECUTE
* 0x00000020 - PAGE_EXECUTE_READ
* 0x00000040 - PAGE_EXECUTE_READWRITE
* 0x00000080 - PAGE_EXECUTE_WRITECOPY
* 0x00000100 - PAGE_GUARD
* 0x00000200 - PAGE_NOCACHE
* 0x00000400 - PAGE_WRITECOMBINE
* 0x00001000 - MEM_COMMIT
* 0x00002000 - MEM_RESERVE
* 0x00010000 - MEM_FREE
* 0x00020000 - MEM_PRIVATE
* 0x00040000 - MEM_MAPPED
* 0x01000000 - MEM_IMAGE
* 0x40000000 - PAGE_TARGETS_NO_UPDATE

A memory map, if its total boolean is true, MUST contain information about all blocks of memory that is allocated at the time of the exception in any way (including, but not limited to, stack, heap, code, data and mapped files).

For the flags, the rules specified in [MEMINFO] and [MEMPROTC] for the bits each one defines MUST be met.
The OID for specifying a value of this type is 1.3.6.1.4.1.37476.9000.46.2.19.7.

4. The referencesOthers boolean

The referencesOthers boolean in an ExceptionInformationRecord MUST properly indicate whether it references other ExceptionInformationRecord structures in the same ASN.1 transmission by using the indices of those records within the transmission.

5. Security Considerations

Implementations MUST assure server safety in the event of exceptions and send valid structures according to the ASN.1 Basic Encoding Rules [X.690].

6. IANA Considerations

A new MIME type for AXIF may need to be registered.

7. References

7.1. Normative References


7.2. Informative References

[X86CTX] NirSoft, "struct CONTEXT",
https://www.nirsoft.net/kernel_struct/vista/CONTEXT.html

Appendix A. ASN.1 Module

This module is written in 2002 ASN.1 [X.680].

AXIF { iso(1) identified-organization(3) dod(6) internet(1) private(4)
    enterprises(1) 37476 freeoid(9000) 46 standard(2) axif(19) module(4) }

DEFINITIONS IMPLICIT TAGS ::= 

BEGIN

-- EXPORTS ALL --

IMPORTS AlgorithmIdentifier, Certificate FROM PKIX1Explicit88 { iso(1)
    identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18) }

id-axif OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6)
    internet(1) private(4) enterprises(1) 37476 freeoid(9000) 46
    standard(2) axif(19) }

AXIF ::= SEQUENCE {
    info SEQUENCE OF ExceptionInformation,
    signature DigitalSignature
}

DigitalSignature ::= SEQUENCE {
    signatureAlgorithm AlgorithmIdentifier,
    -- first certificate MUST sign the message
    certificates [0] SEQUENCE OF Certificate,
    signature BIT STRING
}

ExceptionInformation ::= SEQUENCE OF ExceptionInformationRecord

ExceptionInformationRecord ::= SEQUENCE {
    type OBJECT IDENTIFIER,
    referencesOthers BOOLEAN DEFAULT FALSE
    value ANY DEFINED BY type
}

-- Specifying a NTSTATUS value in an exception information record

id-axif-ntstatus OBJECT IDENTIFIER ::= { id-axif ntstatus(1) }

NTSTATUS ::= INTEGER (-2147483648..2147483647)
-- Specifying a level in an exception information record

id-axif-level OBJECT IDENTIFIER ::= { id-axif level(2) }

-- internal is for internal issues
-- involvesClients is for errors that involve clients (for instance,
-- sharing violations between a client and a server)
-- direct is in the case a file the user requests was not found in the
-- server

AXIFLevel ::= ENUMERATED { internal, involvesClients, direct }

-- Specifying basic information in an exception information record

id-axif-basicInformation OBJECT IDENTIFIER ::= { id-axif
  basicInformation(3) }

BasicInformation ::= SEQUENCE {
  client BOOLEAN DEFAULT FALSE,
  data String
}

String ::= CHOICE {
  ia5String IA5String,
  utf8String UTF8String,
  printableString PrintableString,
  bmpString BMPString
}

-- Specifying a x86 context in an exception information record

id-axif-x86Context OBJECT IDENTIFIER ::= { id-axif
  x86Context(5) }

X86Context ::= OCTET STRING (SIZE(204 | 716 | 720))

-- Specifying a Win32 exception record in an exception information
-- record

id-axif-win32ExceptionRecord OBJECT IDENTIFIER ::= { id-axif
  win32ExceptionRecord(6) }

Win32ExceptionRecord ::= OCTET STRING (SIZE(80))

-- Specifying a memory map in an exception information record

id-axif-memoryMap OBJECT IDENTIFIER ::= { id-axif
  memoryMap(7) }

MemoryMap ::= CHOICE {
  partialMap SET OF MemoryMapEntry,
  totalMap [0] SET OF MemoryMapEntry
}
MemoryMapEntry ::= SEQUENCE {
    address INTEGER(0..MAX),
    size INTEGER(1..MAX),
    protection INTEGER,
    filepath [0] EXPLICIT String OPTIONAL,
    type [1] MemoryType OPTIONAL
}

MemoryType ::= ENUMERATED { stack(0) heap(1) managed-stack(2)
    managed-heap(3) other-stack(4) other-heap(5) }

END

Appendix B. Reference Implementation

This implementation captures exceptions using a vectored exception
handler and fills in AXIF records containing a NTSTATUS (Section 3.1),
a level (Section 3.2), a basic information structure (Section 3.3),
a x86 context (Section 3.5) and a Win32 exception record (Section 3.6).

; Writes an ever-growing AXIF structure to memory, and provides an API
; for accessing it.
; This was created as a reference implementation for AXIF. Backlink:

; This DLL is written in x86 NASM assembly.
; For assembling, NASM (http://www.nasm.us) or a backwards-compatible
; extension of it is required. Follows an example of a command to
; assemble the DLL.
; nasm -f bin -o axiftest.dll axiftest.asm
; To avoid linking hassle, this already contains all necessary headers,
; including the MZ and PE headers.

; This works in Windows XP and later versions as of this writing.
; This uses the undocumented Native API, as well as undocumented
; fields of internal structures. Unofficial documentation in these
; undocumented APIs is at https://undocumented.ntinternals.net/, that
; includes most of the functions imported from NTDLL in the reference
; implementation. The remaining functions are mostly C runtime functions
; (memcpy, memset, wcstoul, wcslen, _vsnwprintf, strtol, ...).

; Windows is a registered trademark of Microsoft Corporation, and Win32
; may be a registered trademark or trademark of Microsoft Corporation in
; some jurisdictions.

; Version 1.0.1.3 (major.minor.build.revision)
; Increment major if incompatible changes done.
; Increment minor if significant, compatible changes done.
; Increment build if building the binary from the source code.
; Increment revision with every revision to the source code.
BITS 32
org 0x10330000

;;; <property>87102750-B7E7-11E6-9598-0800200C9A66: 4</property>
%define RVA(x) (x - $$)
;;; <property>87102750-B7E7-11E6-9598-0800200C9A66: 4</property>
%define HEADERS_START $$
;;; <property>87102750-B7E7-11E6-9598-0800200C9A66: 4</property>
%define IMAGE_START $$
%define runto(x, y) times x-($-$$) db y

; DOS header
dw 0x5A4D
dw 0x90
dd 3
dd 4
dd 0x0000FFFF
dq 0x0B8
dq 0x40
dq 0, 0, 0
dd 0
dd 0x80
db 0x0E, 0x1F, 0xBA, 0x0E, 0x00, 0xB4, 0x09, 0xCD, 0x21, 0xB8, 0x01
db 0x4C, 0xCD, 0x21
db "This is not a program.", 0

runto(128, 0)

; NT header
dd 0x4550
dw 332
dw 3
dd -1
dd 0
dd 0
dw 224
dw 0x2122
dw 267
db 1
db 0
dd CODE_END-CODE_START
dd DATA_END-DATA_START
dd 0
dd RVA(EntryPoint)
dd 0x1000
dd RVA(DATA_START)
dd 0x10330000
dd 0x1000
dd 512
Internet-Draft    Advanced Exception Information Format         May 2017

dw 5,1,1,0,5,1
dd 0
dd IMAGE_END-IMAGE_START
dd HEADERS_END-HEADERS_START
dd 0
dw 3
dw 0x8100
dd 0x00200000
dd 0x00010000
dd 0x00180000
dd 0x00010000
dd 0
dd 0x0010
dd RVA(ExportTable)
dd EXPORTS_END-ExportTable
dd RVA(ImportTable)
dd IMPORTS_END-ImportTable
times 16 dd 0
dd RVA(LoadConfig)
dd 64
times 40 db 0
db ".text", 0, 0, 0
dd CODE_END-CODE_START
dd 0x1000
dd CODE_END-CODE_START
dd 0x1000
dd 0
dd 0
dd 0
dd 0x60C00020
db ".data", 0, 0, 0
dd DATA_END-DATA_START
dd RVA(DATA_START)
dd DATA_END-DATA_START
dd RVA(DATA_START)
dd 0
dd 0
dd 0
dd 0xC0C00040
db ".idata", 0, 0
dd IDATA_END-IDATA_START
dd RVA(IDATA_START)
dd IDATA_END-IDATA_START
dd RVA(IDATA_START)
dd 0
dd 0
dd 0
dd 0xC0C00040
Internet-Draft    Advanced Exception Information Format         May 2017

; End of headers.
align 512, db 0
HEADERS_END:

align 4096, db 0

; [[section .text]]
CODE_START:

ExportTable:
dd 0
dd -1
dd 0
dd 0x1028
dd 0
dd 2
dd 2
dd RVA(axiftest_functions)
dd RVA(axiftest_names)
dd RVA(axiftest_ordinals)

db "axiftest.dll", 0, 0, 0, 0

axiftest_functions:
dd RVA(GetAXIFLog)
dd RVA(GetAXIFLogSize)
; compatibility with naive implementations
dd 0

axiftest_names:
dd RVA(name1)
dd RVA(name2)
; compatibility with naive implementations
dd 0

axiftest_ordinals:
dw 0, 1, -1, 0

name1 db "GetAXIFLog", 0
name2 db "GetAXIFLogSize", 0

EXPORTS_END:
align 16, db 0

LoadConfig:
dd 72
times 14 dd 0
dd __security_cookie
dd handler_table
dd 0 ; number of SE handlers.
handler_table:
; Addresses of SE handlers go here.
; Example (if handler had label Handler1): dd RVA(Handler1)

times 32 db 0x90

__security_check_cookie:
    mov ecx, [ebp-0x04]
    xor ecx, ebp
    cmp ecx, [__security_cookie]
    jnz .buffer_overflow
    mov ecx, [ebp-0x08]
    xor ecx, [ebp+0x04]
    cmp ecx, [__security_cookie+4]
    jz .return
    .buffer_overflow:
    push dword 0xC0000409
    push byte -1
    call dword [ntdll_NtTerminateProcess]
    jmp short .buffer_overflow
    .return:
    ret

align 16

EntryPoint:
    mov eax, [esp+0x08]
    test eax, eax
    jz .terminate
    dec eax
    jnz .return2
    push dword VectoredHandler
    push byte +1
    call dword [ntdll_RtlAddVectoredExceptionHandler]
    mov [excepthandler], eax
    test eax, eax
    jz .return
    push byte +0
    push byte +0
    push byte +0
    push byte +0
    push dword 0x00040062
    call dword [ntdll_RtlCreateHeap]
    mov [axifheap], eax
    testne al
    movzx eax, al
    jmp short .return
.terminate:
push dword [excepthandler]
call dword [ntdll_RtlRemoveVectoredExceptionHandler]
.return2:
mov al, 1
.return:
ret 0x000C

align 16

; Skeleton to build an AXIF record
; Offset Size Element to insert
; 0x0008 0x0004 NTSTATUS
; 0x0045 0x0050 Win32 exception record
; 0x00AD 0x02D0 X86 context
; 0x037E 0x0001 Version string length (in bytes) + 23
; 0x0390 0x0001 Version string length (in bytes) + 5
; 0x0395 0x0001 Version string length (in bytes)
; 0x0396 varies Version string
skeleton:
db 0x30, 0x82, 0x00, 0x00, 0x30, 0x16, 0x06, 0x0E
db 0x2B, 0x06, 0x01, 0x04, 0x00, 0x30, 0x82, 0x0E

; Offset Size Element to insert
; 0x037E 0x0001 Version string length (in bytes) + 23
; 0x0390 0x0001 Version string length (in bytes) + 5
; 0x0395 0x0001 Version string length (in bytes)
; 0x0396 varies Version string
skeleton:
db 0x30, 0x82, 0x00, 0x00, 0x30, 0x16, 0x06, 0x0E
db 0x2B, 0x06, 0x01, 0x04, 0x00, 0x30, 0x82, 0x0E
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

align 16, db 0

; int GetAXIFLogSize()
; Gets the size of the stored AXIF log.
GetAXIFLogSize:
    mov eax, [axiflogsize]
    test eax, eax
    jz .zero
    bsr ecx, eax
    cmp ecx, byte +7
    jge .g1
    dec ecx
    dec ecx
    ret
Internet-Draft    Advanced Exception Information Format         May 2017

.gl:
sar ecx, 3
lea eax, [eax+ecx+3]
bsr ecx, eax
cmp ecx, byte +7
jge .g2
dec ecx

.g2:
sar ecx, 3
lea eax, [eax+ecx+3]
{return:}
ret

Zero:
; If the log is empty, the size is of a SEQUENCE containing an empty
; SEQUENCE (30 02 30 00), or 4 bytes.
mov al, 4
jmp short .return

align 16

; void __stdcall GetAXIFLog(void* buf)
; Fills a buffer of size GetAXIFLogSize() bytes with an AXIF containing
; all exceptions that occurred.
GetAXIFLog:
push ebp
mov ebp, esp
push edi
mov edi, [ebp+0x08]
cmp dword [axiflog], byte +0
jnz .haslog
; No log, so fill it with 30 02 30 00 (little endian 0x00300230)
mov eax, [ebp+0x08]
mov dword [eax], 0x00300230
{return:}
pop ebp
ret 0x0004

.haslog:
call GetAXIFLogSize
mov edx, [axiflogsize]
sub eax, edx
mov ecx, eax
mov al, 0x30
stosb
lea edi, [edi+ecx-1]
push edi
std
bsr ecx, edx
and cl, 0xF8
push ecx
.loop1:
    mov eax, edx
    shr eax, cl
    stosb
    sub cl, 8
    jnz .loop1
    pop eax
    shr eax, 3
    lea edx, [edx+ecx+3]
    or al, 0x80
    stosb
    mov al, 0x30
    stosb
    bsr ecx, edx
    and cl, 0xF8
    push ecx
    .loop2:
    mov eax, edx
    shr eax, cl
    stosb
    sub cl, 8
    jnz .loop2
    pop eax
    shr eax, 3
    lea edx, [edx+ecx+3]
    or al, 0x80
    stosb
    cid
    pop edi
    inc edi
    push dword [axiflogsize]
    push dword [axiflog]
    push edi
    call dword [ntdll_memcpy]
    jmp short .return

align 16

; This is the function that actually creates the AXIF records, with an
; NTSTATUS value, a level (set to internal), basic information with
; client=true and the data string containing the Windows version,
; a Win32 exception record and a x86 context.
VectoredHandler:
    push ebp
    mov ebp, esp
    push ebx
    push esi
    mov ebx, [axiflog]
    test ebx, ebx
    jnz .haslog
mov eax, [fs:0x0030]
push dword [eax+0xAC]
push dword [eax+0xA8]
push dword [eax+0xA4]
push esp
push dword bi_format
push byte +0x2F
lea eax, [ebp-0x30]
push dword versionstring
call dword [ntdll_vsnprintf]
push dword versionstring
call dword [ntdll_strlen]
mov [versionstringlen], al
add eax, 0x00000396
mov [recordsize], eax
mov [axiflogsize], eax
push eax
push byte +0
push dword [axifheap]
call dword [ntdll_RtlAllocateHeap]
xor ebx, ebx
jmp short .allocated
.haslog:
mov ebx, [axiflogsize]
mov eax, ebx
add eax, [recordsize]
mov [axiflogsize], eax
push eax
push dword [axiflog]
push byte +0
push dword [axifheap]
call dword [ntdll_RtlReAllocateHeap]
.allocated:
mov [axiflog], eax
add ebx, eax
push dword 0x00000396
push dword skeleton
push ebx
call dword [ntdll_memcpy]
mov esi, [ebp+0x08]
mov eax, [esi]
mov eax, [eax]; NTSTATUS gets set to exception code.
bswap eax; ASN.1 uses big-endian, so swap bytes.
mov [ebx+0x18], eax; Write the NTSTATUS.
; Copy the Win32 exception record.
push byte +0x50
push dword [esi]
lea eax, [ebx+0x45]
push eax
call dword [ntdll_memcpy]
; Win32 exception record is little endian, no need for byte swap.
; Copy the x86 context.
push dword 0x000002D0
push dword [esi+0x04]
lea eax, [ebx+0x00AD]
push eax
call dword [ntdll_memcpy]
; x86 context is little endian, no need for byte swap.
; Set lengths.
mov eax, [recordsize]
sub eax, byte +4
mov [ebx+0x002], ax
mov eax, [versionstringlen]
push eax
mov [ebx+0x00395], al
add al, 5
mov [ebx+0x00390], al
add al, 18 ; 23 - 5 = 18
mov [ebx+0x0037E], al
push dword versionstring
lea eax, [ebx+0x00396]
push eax
call dword [ntdll_memcpy]
mov esi, [ebp-0x08]
mov ebx, [ebp-0x04]
xor eax, eax
mov esp, ebp
pop ebp
ret 0x0004

align 4096, db 0
CODE_END:
; section .data
DATA_START:
; DLL data.

__security_cookie dq 0 ; 8-byte security cookie, safer

; Variables for the reference implementation.
axiflog dd 0
; This starts at 0, as the size of the two SEQUENCEs is automatically
; added to the return value of GetAXIFLogSize when it is called.
axiflogsize dd 0

; Contains the version string (e.g. "Windows NT 6.3.9600") that will
; be put in the data field of the BasicInformation (Section 3.3) as
; a UTF8String.
versionstring times 48 db 0
recordsize dd 0
axifheap dd 0
versionstringlen dd 0
excepthandler dd 0
; The import table is expanded to include any Native API functions that
; may be useful in expanding the reference implementation with new
; features.

;; <property>8F346490-B7E8-11E6-9598-0800200C9A66: 8</property>
ImportTable:
   dd RVA(ntdll_LookupTable)
   dd 0
   dd 0
   dd RVA(ntdll_string)
   dd RVA(ntdll_imports)
   dd 0
   dd 0
   dd 0
   dd 0
   dd 0

;; <property>8F346490-B7E8-11E6-9598-0800200C9A66: 9</property>
ntdll_LookupTable:
   dd RVA(ntdll_import1)
   dd RVA(ntdll_import2)
   dd RVA(ntdll_import3)
   dd RVA(ntdll_import4)
   dd RVA(ntdll_import5)
   dd RVA(ntdll_import6)
   dd RVA(ntdll_import7)
   dd RVA(ntdll_import8)
   dd RVA(ntdll_import9)
   dd RVA(ntdll_import10)
   dd RVA(ntdll_import11)
   dd RVA(ntdll_import12)
   dd RVA(ntdll_import13)
   dd RVA(ntdll_import14)
   dd RVA(ntdll_import15)
   dd RVA(ntdll_import16)
   dd RVA(ntdll_import17)
   dd RVA(ntdll_import18)
   dd RVA(ntdll_import19)
   dd RVA(ntdll_import20)
   dd RVA(ntdll_import21)
   dd RVA(ntdll_import22)
   dd RVA(ntdll_import23)
   dd RVA(ntdll_import24)
   dd RVA(ntdll_import25)
   dd RVA(ntdll_import26)
   dd RVA(ntdll_import27)
   dd RVA(ntdll_import28)
dd RVA(ntdll_Import29)
dd RVA(ntdll_Import30)
dd RVA(ntdll_Import31)
dd RVA(ntdll_Import32)
dd RVA(ntdll_Import33)
dd RVA(ntdll_Import34)
dd RVA(ntdll_Import35)
dd RVA(ntdll_Import36)
dd RVA(ntdll_Import37)
dd 0

;;; <property>8F346490-B7E8-11E6-9598-0800200C9A66: 10</property>
ntdll_imports:

; Imports from NTDLL.DLL
ntdll_memmove dd RVA(ntdll_Import1)
ntdll_RtlEnterCriticalSection dd RVA(ntdll_Import2)
ntdll_RtlLeaveCriticalSection dd RVA(ntdll_Import3)
ntdll_NtTerminateProcess dd RVA(ntdll_Import4)
ntdll_vsnwprintf dd RVA(ntdll_Import5)
ntdll_memset dd RVA(ntdll_Import6)
ntdll_RtlInitializeCriticalSectionAndSpinCount dd RVA(ntdll_Import7)
ntdll_RtlDeleteCriticalSection dd RVA(ntdll_Import8)
ntdll_wcstoul dd RVA(ntdll_Import9)
ntdll_RtlAllocateHeap dd RVA(ntdll_Import10)
ntdll_memcpy dd RVA(ntdll_Import11)
ntdll_RtlFreeHeap dd RVA(ntdll_Import12)
ntdllwcslen dd RVA(ntdll_Import13)
ntdll_strtol dd RVA(ntdll_Import14)
ntdll_NtAllocateVirtualMemory dd RVA(ntdll_Import15)
ntdll_NtCreateFile dd RVA(ntdll_Import16)
ntdll_NtWriteFile dd RVA(ntdll_Import17)
ntdll_NtInitUnicodeString dd RVA(ntdll_Import18)
ntdll_NtClose dd RVA(ntdll_Import19)
ntdll_NtDeleteFile dd RVA(ntdll_Import20)
ntdll_NtFreeVirtualMemory dd RVA(ntdll_Import21)
ntdll_RtlGetCurrentDirectory_U dd RVA(ntdll_Import22)
ntdll_NtCreateThread dd RVA(ntdll_Import23)
ntdll_NtWaitForSingleObject dd RVA(ntdll_Import24)
ntdll_NtTerminateThread dd RVA(ntdll_Import25)
ntdll_NtCreateEvent dd RVA(ntdll_Import26)
ntdll_NtQuerySystemInformation dd RVA(ntdll_Import27)
ntdll_NtSetEvent dd RVA(ntdll_Import28)
ntdll_ltolw dd RVA(ntdll_Import29)
ntdll_RtlExitUserThread dd RVA(ntdll_Import30)
ntdll_RtlUnwind dd RVA(ntdll_Import31)
ntdll_RtlAddVectoredExceptionHandler dd RVA(ntdll_Import32)
ntdll_RtlRemoveVectoredExceptionHandler dd RVA(ntdll_Import33)
ntdll_vsnprintf dd RVA(ntdll_Import34)
ntdll_strlen dd RVA(ntdll_Import35)
ntdll_RtlReAllocateHeap dd RVA(ntdll_Import36)
ntdll_RtlCreateHeap dd RVA(ntdll_Import37)
dd 0

;;; <property>8F346490-B7E8-11E6-9598-0800200C9A66: 11</property>
ntdll_Import1:
dw 2150
db "memmove", 0
ntdll_Import2:
dw 874
db "RtlEnterCriticalSection", 0
ntdll_Import3:
dw 1114
db "RtlLeaveCriticalSection", 0
ntdll_Import4:
dw 579
db "NtTerminateProcess", 0
ntdll_Import5:
dw 2095
db "_vsnwprintf", 0
ntdll_Import6:
dw 2152
db "memset", 0
ntdll_Import7:
dw 1036
db "RtlInitializeCriticalSectionAndSpinCount", 0
ntdll_Import8:
dw 825
db "RtlDeleteCriticalSection", 0
ntdll_Import9:
dw 2217
db "wcstoul", 0
ntdll_Import10:
dw 691
db "RtlAllocateHeap", 0
ntdll_Import11:
dw 2148
db "memcpy", 0
ntdll_Import12:
dw 938
db "RtlFreeHeap", 0
ntdll_Import13:
dw 2203
db "wcslen", 0
ntdll_Import14:
dw 2181
db "strtol", 0
ntdll_Import15:
dw 201
db "NtAllocateVirtualMemory", 0
ntdll_Import16:
dw 253
db "NtCreateFile", 0
ntdll_Import17:
dw 628
db "NtWriteFile", 0
ntdll_Import18:
dw 1028
db "RtlInitUnicodeString", 0
ntdll_Import19:
dw 236
db "NtClose", 0
ntdll_Import20:
dw 296
db "NtDeleteFile", 0
ntdll_Import21:
dw 330
db "NtFreeVirtualMemory", 0
ntdll_Import22:
dw 956
db "RtlGetCurrentDirectory_U", 0
ntdll_Import23:
dw 276
db "NtCreateThread", 0
ntdll_Import24:
dw 604
db "NtWaitForSingleObject", 0
ntdll_Import25:
dw 580
db "NtTerminateThread", 0
ntdll_Import26:
dw 251
db "NtCreateEvent", 0
ntdll_Import27:
dw 462
db "NtQuerySystemInformation", 0
ntdll_Import28:
dw 527
db "NtSetEvent", 0
ntdll_Import29:
dw 2061
db "_ltow", 0
ntdll_Import30:
dw 896
db "RtlExitUserThread", 0
ntdll_Import31:
dw 1379
db "RtlUnwind", 0
ntdll_Import32:
dw 684
db "RtlAddVectoredExceptionHandler", 0
Internet-Draft    Advanced Exception Information Format         May 2017

ntdll_Import33:
dw 1244
db "RtlRemoveVectoredExceptionHandler", 0
ntdll_Import34:
dw 2093
db "_vsnprintf", 0
ntdll_Import35:
dw 2169
db "strlen", 0
ntdll_Import36:
dw 1221
db "RtlReAllocateHeap", 0
ntdll_Import37:
dw 785
db "RtlCreateHeap", 0

ntdll_string db "ntdll.dll", 0

IMPORTS_END:
align 4096, db 0
IDATA_END:

; End of image.
IMAGE_END.

; The reference implementation ends here.

Author’s Address

Luis Camara (@luis140219 on GitHub)

EMail: luis.camara@live.com.pt