Api set resolution

lucasg.github.io/2017/10/15/Api-set-resolution

15 Oct 2017

Windows API Sets schema is a weird dll redirection mechanism Microsoft introduced in Win7 (and perfected in Win8.1) but no one really know why it's useful. What's more there is little to none official documentation on the subject (and it's not even up to date) which makes me think MS don't really want to officially support the schema.

All I know (before writing this post) is to solve a missing api min-win, you usually rely on copying the whole redist folder when deploying a statically compiled binary and calling it a day :

```
def _CopyUCRTRuntime(target_dir, source_dir, target_cpu, dll_pattern, suffix):
  """Copy both the msvcp and vccorlib runtime DLLs, only if the target doesn't
  exist, but the target directory does exist."""
  for file_part in ('msvcp', 'vccorlib', 'vcruntime'):
    dll = dll_pattern % file_part
    target = os.path.join(target_dir, dll)
    source = os.path.join(source_dir, dll)
    _CopyRuntimeImpl(target, source)
  # Copy the UCRT files needed by VS 2015 from the Windows SDK. This location
  # includes the api-ms-win-crt-*.dll files that are not found in the Windows
  # directory. These files are needed for component builds.
  # If WINDOWSSDKDIR is not set use the default SDK path. This will be the case
  # when DEPOT_TOOLS_WIN_TOOLCHAIN=0 and vcvarsall.bat has not been run.
  win_sdk_dir = os.path.normpath(
      os.environ.get('WINDOWSSDKDIR',
                     'C:\\Program Files (x86)\\Windows Kits\\10'))
  ucrt_dll_dirs = os.path.join(win_sdk_dir, r'Redist\ucrt\DLLs', target_cpu)
  ucrt_files = glob.glob(os.path.join(ucrt_dll_dirs, 'api-ms-win-*.dll'))
  assert len(ucrt_files) > 0
  for ucrt_src_file in ucrt_files:
    file_part = os.path.basename(ucrt_src_file)
    ucrt_dst_file = os.path.join(target_dir, file_part)
    _CopyRuntimeImpl(ucrt_dst_file, ucrt_src_file, False)
  _CopyRuntimeImpl(os.path.join(target_dir, 'ucrtbase' + suffix),
                   os.path.join(source_dir, 'ucrtbase' + suffix))
```

Can you find which application does that ?

Fortunately, third-party research covers pretty much everything you need to know the subject. However, I had to write an api set resolver for the Dependencies application I've been working on this year. Following in this post is how the NT loader resolve an api set library.

Api sets dll are "virtual libraries" actually implementing only contract APIs which will be resolved by the NT loader. According to Ionescu's Esoteric Hooks presentation the official aim is to decouple 'API' features (heap allocations, strings manipulations, etc.) from subsystem implementations (enumerating processes, registering services, etc.). I'm pretty sure this is also another step into getting rid of the undying cockroach that is the Win32 subsystem in Windows desktops. Anyway, api set schema is probably what made UWP apps possible :

Patent Application Publication Dec. 11, 2014 Sheet 1 of 12 US 2014/0366045 A1

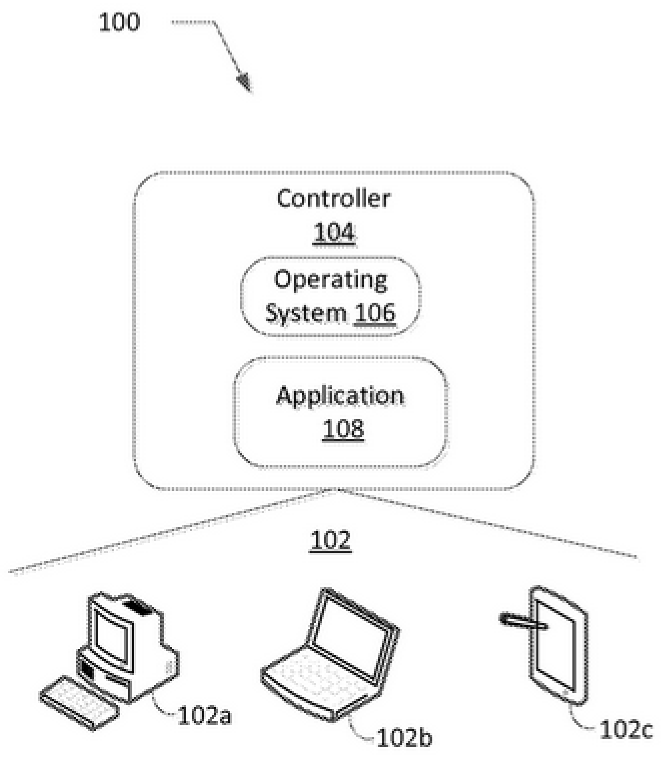
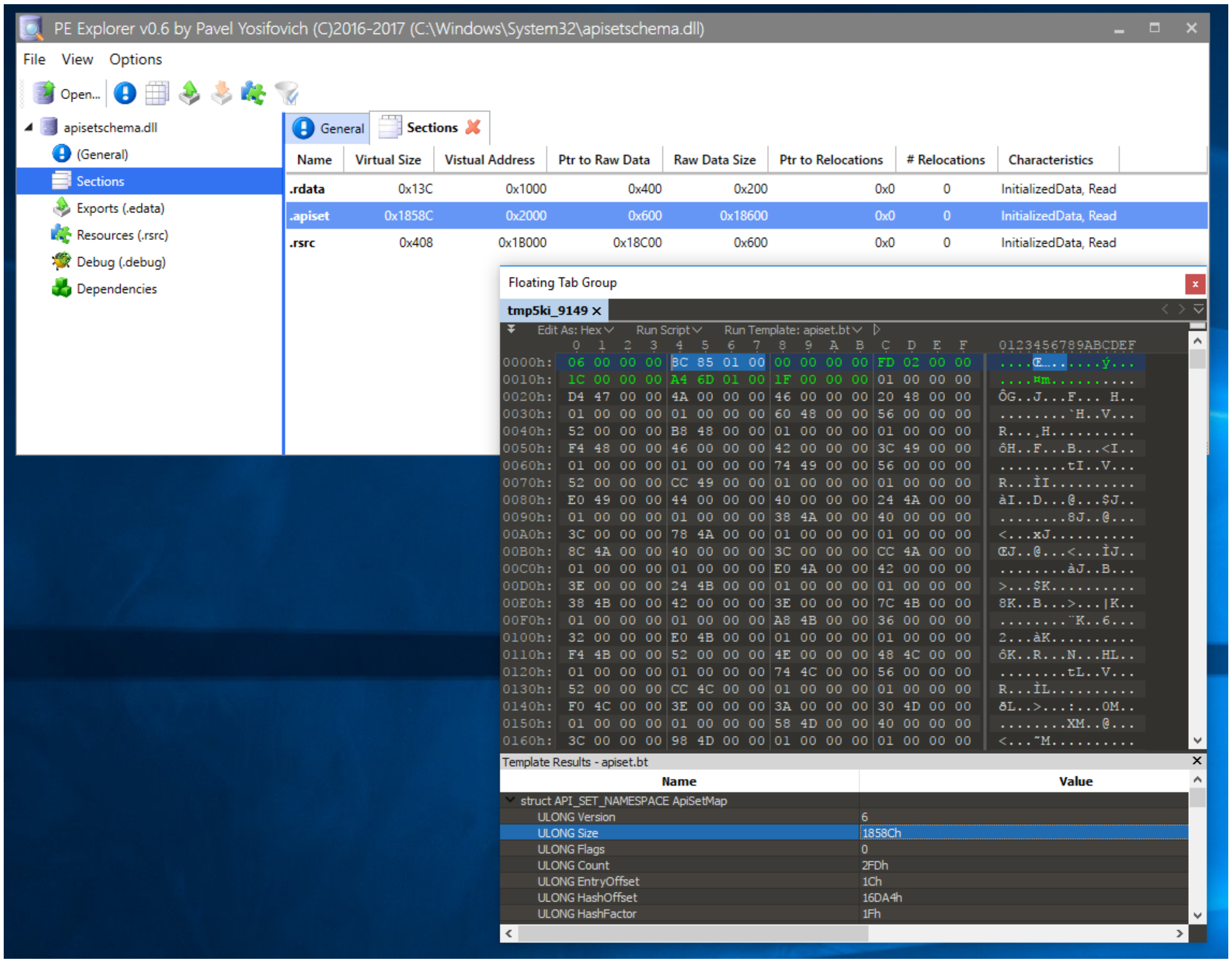


FIG. 1A

As said previously, the api set schema (and its evolution along Windows versions) has already been reversed and publicly documented, so I'll just do a quick recap on it's actual implementation.

`%Windir%\System32\apisetschema.dll` is the dll implementing the contract resolution between virtual api-min-win dlls and host libraries in its `.apiset` section :


PE Explorer v0.6 by Pavel Yosifovich (C)2016-2017 (C:\Windows\System32\apisetschema.dll)

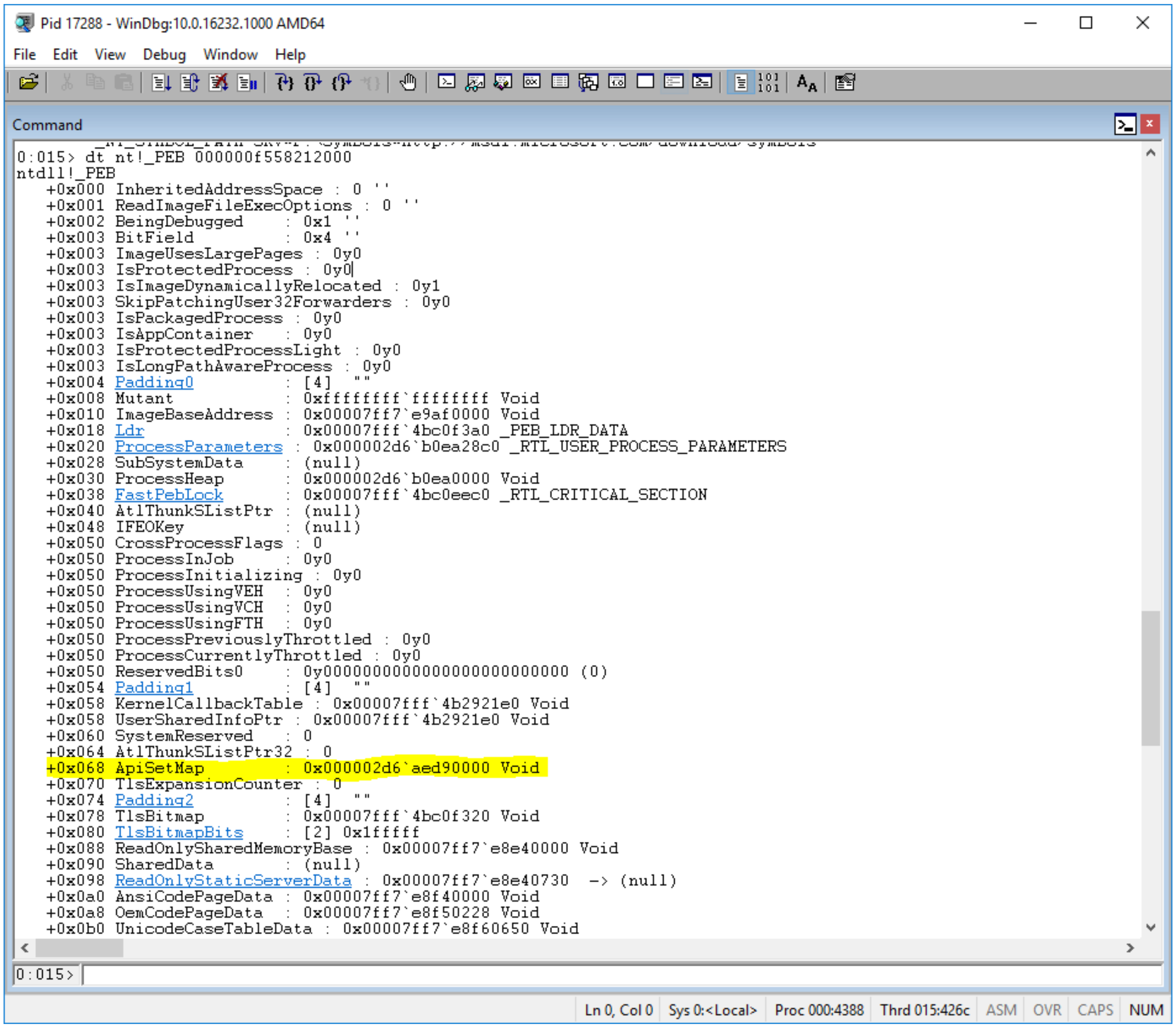
Name	Virtual Size	Vistual Address	Ptr to Raw Data	Raw Data Size	Ptr to Relocations	# Relocations	Characteristics
.rdata	0x13C	0x1000	0x400	0x200	0x0	0	InitializedData, Read
.apiset	0x1858C	0x2000	0x600	0x18600	0x0	0	InitializedData, Read
.rsrc	0x408	0x1B000	0x18C00	0x600	0x0	0	InitializedData, Read

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 06 00 00 00 BC 85 01 00 00 00 00 00 FD 02 00 00  ....Œ.......ý...
0010h: 1C 00 00 00 A4 6D 01 00 1F 00 00 00 01 00 00 00  ....¤m..........
0020h: D4 47 00 00 4A 00 00 00 46 00 00 00 20 48 00 00  ÔG..J...F... H..
0030h: 01 00 00 00 01 00 00 00 60 48 00 00 56 00 00 00  ........`H..V...
0040h: 52 00 00 00 B8 48 00 00 01 00 00 00 01 00 00 00  R...¸H..........
0050h: F4 48 00 00 46 00 00 00 42 00 00 00 3C 49 00 00  ôH..F...B...<I..
0060h: 01 00 00 00 01 00 00 00 74 49 00 00 56 00 00 00  ........tI..V...
0070h: 52 00 00 00 CC 49 00 00 01 00 00 00 01 00 00 00  R...ÌI..........
0080h: E0 49 00 00 44 00 00 00 40 00 00 00 24 4A 00 00  àI..D...@...$J..
0090h: 01 00 00 00 01 00 00 00 38 4A 00 00 40 00 00 00  ........8J..@...
00A0h: 3C 00 00 00 78 4A 00 00 01 00 00 00 01 00 00 00  <...xJ..........
00B0h: 8C 4A 00 00 40 00 00 00 3C 00 00 00 CC 4A 00 00  ŒJ..@...<...ÌJ..
00C0h: 01 00 00 00 01 00 00 00 E0 4A 00 00 42 00 00 00  ........àJ..B...
00D0h: 3E 00 00 00 24 4B 00 00 01 00 00 00 01 00 00 00  >...$K..........
00E0h: 38 4B 00 00 42 00 00 00 3E 00 00 00 7C 4B 00 00  8K..B...>...|K..
00F0h: 01 00 00 00 01 00 00 00 A8 4B 00 00 36 00 00 00  ........¨K..6...
0100h: 32 00 00 00 E0 4B 00 00 01 00 00 00 01 00 00 00  2...àK..........
0110h: F4 4B 00 00 52 00 00 00 4E 00 00 00 48 4C 00 00  ôK..R...N...HL..
0120h: 01 00 00 00 01 00 00 00 74 4C 00 00 56 00 00 00  ........tL..V...
0130h: 52 00 00 00 CC 4C 00 00 01 00 00 00 01 00 00 00  R...ÌL..........
0140h: F0 4C 00 00 3E 00 00 00 3A 00 00 00 30 4D 00 00  ðL..>...:...0M..
0150h: 01 00 00 00 01 00 00 00 58 4D 00 00 40 00 00 00  ........XM..@...
0160h: 3C 00 00 00 98 4D 00 00 01 00 00 00 01 00 00 00  <...˜M..........
```

Template Results - apiset.bt

Name	Value
struct API_SET_NAMESPACE ApiSetMap	
ULONG Version	6
ULONG Size	1858Ch
ULONG Flags	0
ULONG Count	2FDh
ULONG EntryOffset	1Ch
ULONG HashOffset	16DA4h
ULONG HashFactor	1Fh

This section is actually present in every process via the PEB (probably using a COW mechanism). :


Pid 17288 - WinDbg:10.0.16232.1000 AMD64
File Edit View Debug Window Help
Command

```
0:015> dt nt!_PEB 000000f558212000
ntdll!_PEB
   +0x000 InheritedAddressSpace : 0 ''
   +0x001 ReadImageFileExecOptions : 0 ''
   +0x002 BeingDebugged    : 0x1 ''
   +0x003 BitField         : 0x4 ''
   +0x003 ImageUsesLargePages : 0y0
   +0x003 IsProtectedProcess : 0y0
   +0x003 IsImageDynamicallyRelocated : 0y1
   +0x003 SkipPatchingUser32Forwarders : 0y0
   +0x003 IsPackagedProcess : 0y0
   +0x003 IsAppContainer   : 0y0
   +0x003 IsProtectedProcessLight : 0y0
   +0x003 IsLongPathAwareProcess : 0y0
   +0x004 Padding0         : [4]  ""
   +0x008 Mutant           : 0xffffffff`ffffffff Void
   +0x010 ImageBaseAddress : 0x00007ff7`e9af0000 Void
   +0x018 Ldr              : 0x00007fff`4bc0f3a0 _PEB_LDR_DATA
   +0x020 ProcessParameters : 0x000002d6`b0ea28c0 _RTL_USER_PROCESS_PARAMETERS
   +0x028 SubSystemData    : (null)
   +0x030 ProcessHeap      : 0x000002d6`b0ea0000 Void
   +0x038 FastPebLock      : 0x00007fff`4bc0eec0 _RTL_CRITICAL_SECTION
   +0x040 AtlThunkSListPtr : (null)
   +0x048 IFEOKey          : (null)
   +0x050 CrossProcessFlags : 0
   +0x050 ProcessInJob     : 0y0
   +0x050 ProcessInitializing : 0y0
   +0x050 ProcessUsingVEH  : 0y0
   +0x050 ProcessUsingVCH  : 0y0
   +0x050 ProcessUsingFTH  : 0y0
   +0x050 ProcessPreviouslyThrottled : 0y0
   +0x050 ProcessCurrentlyThrottled : 0y0
   +0x050 ReservedBits0    : 0y000000000000000000000000 (0)
   +0x054 Padding1         : [4]  ""
   +0x058 KernelCallbackTable : 0x00007fff`4b2921e0 Void
   +0x058 UserSharedInfoPtr : 0x00007fff`4b2921e0 Void
   +0x060 SystemReserved   : 0
   +0x064 AtlThunkSListPtr32 : 0
   +0x068 ApiSetMap        : 0x000002d6`aed90000 Void
   +0x070 TlsExpansionCounter : 0
   +0x074 Padding2         : [4]  ""
   +0x078 TlsBitmap        : 0x00007fff`4bc0f320 Void
   +0x080 TlsBitmapBits    : [2] 0x1fffff
   +0x088 ReadOnlySharedMemoryBase : 0x00007ff7`e8e40000 Void
   +0x090 SharedData       : (null)
   +0x098 ReadOnlyStaticServerData : 0x00007ff7`e8e40730  -> (null)
   +0x0a0 AnsiCodePageData : 0x00007ff7`e8f40000 Void
   +0x0a8 OemCodePageData  : 0x00007ff7`e8f50228 Void
   +0x0b0 UnicodeCaseTableData : 0x00007ff7`e8f60650 Void
0:015>
```

The `PEB.ApiSetMap` points to a `API_SET_NAMESPACE_V6` (not an official denomination) on Windows 10 :

```
typedef struct {
  ULONG Version;     // v2 on Windows 7, v4 on Windows 8.1  and v6 on Windows 10

  ULONG Size;        // apiset map size (usually the .apiset section virtual size)

  ULONG Flags;       // according to Geoff Chappell,  tells if the map is sealed or
not.

  ULONG Count;       // hash table entry count

  ULONG EntryOffset; // Offset to the api set entries values

  ULONG HashOffset;  // Offset to the api set entries hash indexes

  ULONG HashFactor;  // multiplier to use when computing hash

} API_SET_NAMESPACE;
```

The api set schema namespace is basically a hash table with values being the following structure :

```
// Hash table index (just an optimization "trick")

typedef struct {
  ULONG Hash;
  ULONG Index;
} API_SET_HASH_ENTRY;

// Hash table value

typedef struct {
  ULONG Flags;        // sealed flag in bit 0

  ULONG NameOffset;   // Offset to the ApiSet library name PWCHAR (e.g. "api-ms-win-
core-job-l2-1-1")

  ULONG NameLength;   // Ignored

  ULONG HashedLength; // Apiset library name length

  ULONG ValueOffset;  // Offset the list of hosts library implement the apiset
contract (points to API_SET_VALUE_ENTRY array)

  ULONG ValueCount;   // Number of hosts libraries

} API_SET_NAMESPACE_ENTRY;

// Host Library entry

typedef struct {
  ULONG Flags;        // sealed flag in bit 0

  ULONG NameOffset;   // Offset to the ApiSet library name PWCHAR (e.g. "api-ms-win-
core-job-l2-1-1")

  ULONG NameLength;   // Apiset library name length

  ULONG ValueOffset;  // Offset to the Host library name PWCHAR (e.g. "ucrtbase.dll")

  ULONG ValueLength;  // Host library name length

} API_SET_VALUE_ENTRY;
```

The ApiSet resolution is implemented in 3 non exported functions in the `ntdll` (although there are public symbols present) :

- `ApiSetResolveToHost` for the "high level" resolution (take a `UNICODE_STRING` name and fill out another `UNICODE_STRING` structure)
- `ApiSetpSearchForApiSet` which implement the ApiSetMap hash table retrieval
- `ApiSetpSearchForApiSetHost` which can discriminate dll when an apiset library has several "hosts" libraries (not a frequent case).

`ApiSetpSearchForApiSet` is a pretty traditional hash table getter. Since hash buckets may collide, there is an additional check on the apiset library name returned by the hash table :

```
PAPI_SET_NAMESPACE_ENTRY
__fastcall ApiSetpSearchForApiSet(
  _In_ PAPI_SET_NAMESPACE ApiNamespace,
  _In_ PWCHAR ApiNameToResolve,
  _In_ uint16_t ApiNameToResolveSize
)
{
  __int16 _ApiNameToResolveSize; // si@1

  ULONG HashKey; // er9@1

  const WCHAR *_ApiNameToResolve; // rbp@1

  API_SET_NAMESPACE *ApiNamespacePtr; // r10@1

  PWCHAR pApiNameToResolve; // r11@1

  __int64 _ApiNameToResolveCount; // rbx@2

  WCHAR ApiNameToResolveCurChar; // dx@3

  API_SET_NAMESPACE_ENTRY *FoundEntry; // rbx@6

  int HashCounter; // er8@6

  int ApiSetEntryCount; // ecx@6

  int HashIndex; // edx@7

  signed __int64 HashOffset; // r11@7


  _ApiNameToResolveSize = ApiNameToResolveSize;
  HashKey = 0;
  _ApiNameToResolve = ApiNameToResolve;
  ApiNamespacePtr = ApiNamespace;
  pApiNameToResolve = ApiNameToResolve;

  FoundEntry = NULL;
  HashCounter = 0;
  ApiSetEntryCount = ApiNamespace->Count - 1;

  if (ApiSetEntryCount < 0)
    return NULL;

  if (!ApiNameToResolveSize)
    return NULL;

  // HashKey = Hash(ApiNameToResolve.ToLower())

  _ApiNameToResolveCount = (uint16_t) ApiNameToResolveSize;
  do
```

```
    {
      ApiNameToResolveCurChar = *pApiNameToResolve;
      if ((unsigned __int16)(*pApiNameToResolve - 'A') <= 0x19u) // that's BAD
normalization !

        ApiNameToResolveCurChar += ' ';
      ++pApiNameToResolve;

      HashKey = HashKey * ApiNamespace->HashFactor + ApiNameToResolveCurChar;

      --_ApiNameToResolveCount;
    } while (_ApiNameToResolveCount);


    // Looking for matching hash key

    while (1)
    {
      HashIndex = (ApiSetEntryCount + HashCounter) >> 1;
      HashOffset = ApiNamespacePtr->HashOffset + sizeof(uintptr_t) * HashIndex;
      if (HashKey < ((ULONG *)((uintptr_t) ApiNamespacePtr + HashOffset))[0])
      {
        ApiSetEntryCount = HashIndex - 1;
        goto CHECK_COUNTERS;
      }
      if (HashKey <= ((ULONG *)((uintptr_t) ApiNamespacePtr + HashOffset))[0])
        break;

      HashCounter = HashIndex + 1;

    CHECK_COUNTERS:
      if (HashCounter > ApiSetEntryCount)
        return NULL;
    }

    // Get the corresponding hash bucket value

    FoundEntry = (API_SET_NAMESPACE_ENTRY *)((char *)ApiNamespacePtr
      + sizeof(API_SET_NAMESPACE_ENTRY) * *(ULONG *)((char *)&ApiNamespacePtr->Size +
HashOffset)
      + ApiNamespacePtr->EntryOffset);

    if (!FoundEntry)
      return NULL;

    // Check the returned hash entry actually correspond to the given ApiSet name

    if(0 == RtlCompareUnicodeStrings(
      /* _In_ PWCHAR */ _ApiNameToResolve,
      /* _In_ SHORT  */ _ApiNameToResolveSize,
      /* _In_ PWCHAR */ ((uintptr_t)ApiNamespacePtr + FoundEntry->NameOffset),
      /* _In_ SHORT  */ FoundEntry->HashedLength >> 1, // FoundEntry->HashedLength /
sizeof(WCHAR)

      TRUE              // Ignore case

    )) {
      return FoundEntry;
    }


    return NULL;
  }
```

Since the returned entry may reference multiple hosts library (apparently that's a feature MS devs needed) there is an additional function to return the exact host dll based on a "ParentName" : `ApiSetpSearchForApiSetHost`. Honestly, I always set the `ParentName` to `NULL` and get correct results so there must be a "default" result.

```
  PAPI_SET_VALUE_ENTRY
  __stdcall ApiSetpSearchForApiSetHost(
    _In_ PAPI_SET_NAMESPACE_ENTRY Entry,
    _In_ PWCHAR *ParentName,
    _In_ SHORT ParentNameLen,
    _In_ PAPI_SET_NAMESPACE ApiNamespace
  )
  {
    __int64 _EntryValueOffset; // r12@1

    int Counter; // ebp@1

    API_SET_NAMESPACE *_ApiNamespacePtr; // r15@1

    int EntryAliasCount; // ebx@1

    PWCHAR *_ApiToResolveSource; // r10@1

    API_SET_VALUE_ENTRY *FoundEntry; // rdi@1

    SHORT _ApiToResolveLen; // r13@2

    int EntryAliasIndex; // esi@3

    API_SET_VALUE_ENTRY *AliasEntry; // r14@3

    int _result; // eax@3

    PWCHAR *_ApiToResolve; // [sp+68h] [bp+10h]@1


    _ApiToResolveSource = ParentName;

    _ApiNamespacePtr = ApiNamespace;
    _EntryValueOffset = Entry->ValueOffset;
    FoundEntry = (API_SET_VALUE_ENTRY *)((uintptr_t)ApiNamespace +
_EntryValueOffset);

    // Unique host library entry : bail immediately

    EntryAliasCount = Entry->ValueCount - 1;
    if (EntryAliasCount == 0)
      return FoundEntry;

    Counter = 1;
    _ApiToResolve = ParentName;
    _ApiToResolveLen = ParentNameLen;
    do
    {
      EntryAliasIndex = (EntryAliasCount + Counter) >> 1;
      AliasEntry = (API_SET_VALUE_ENTRY *)((char *)_ApiNamespacePtr
        + sizeof(API_SET_VALUE_ENTRY) * EntryAliasIndex
        + _EntryValueOffset);

      // Compare API_SET_VALUE_ENTRY.NameOffset to ParentName

      _result = RtlCompareUnicodeStrings(
        /* _In_ PWCHAR */ _ApiToResolveSource,
        /* _In_ SHORT  */ _ApiToResolveLen,
        /* _In_ PWCHAR */ ((uintptr_t)_ApiNamespacePtr + AliasEntry->NameOffset),
        /* _In_ SHORT  */ AliasEntry->NameLength >> 1,
        TRUE // IgnoreCase

      );

      if (_result < 0)
      {
        EntryAliasCount = EntryAliasIndex - 1;
      }
      else
      {
        if (_result == 0)
        {
          return (API_SET_VALUE_ENTRY *)((char *)_ApiNamespacePtr
            + sizeof(API_SET_VALUE_ENTRY) * ((EntryAliasCount + Counter) >> 1)
            + _EntryValueOffset);
        }

        Counter = EntryAliasIndex + 1;
      }

      _ApiToResolveSource = _ApiToResolve;
    } while (Counter <= EntryAliasCount);

    return FoundEntry;
  }
```

`ApiSetResolveToHost` is the function wrapping the other previous two in order to "hide" the hash table implementation details from the point a view of a third-party developer (being here a MS dev since none of this mechanism is officially accessible). The only singular points are :

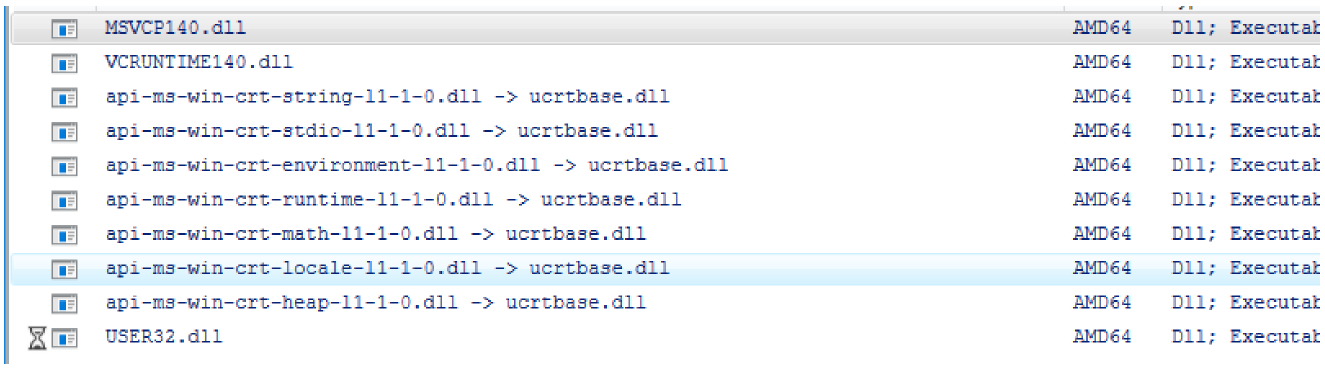
- `ApiSetResolveToHost` checks the apiset library name is actually prefixed by `"api-"` or `"ext-"`
- the apiset library name is truncated before being fed to `ApiSetpSearchForApiSet` : it get rid of the `".dll"` extension (which is a just an application hint) and everything after the last hyphen. That's actually understandable : after the last hyphen is the "build" version, and supporting a strict comparison of apiset library name would make the ApiSet Map size explode.

```
const uint64_t API_ = (uint64_t)0x2D004900500041; // L"api-"

const uint64_t EXT_ = (uint64_t)0x2D005400580045; // L"ext-";


NTSTATUS
__fastcall ApiSetResolveToHost(
  _In_ PAPI_SET_NAMESPACE ApiNamespace,
  _In_ PUNICODE_STRING ApiToResolve,
  _In_ PUNICODE_STRING ParentName,
  _Out_ PBOOLEAN Resolved,
  _Out_ PUNICODE_STRING Output
)
{
  __int64 ApiNamespacePtr; // rdi@1

  char IsResolved; // bl@1

  PBOOLEAN pIsResolved; // r15@1

  UNICODE_STRING *_ParentName; // r14@1

  unsigned __int16 ApiSetNameBufferSize; // cx@1

  wchar_t *ApiSetNameBuffer; // rdx@2

  uint64_t ApiSetNameBufferPrefix; // rax@2

  NTSTATUS Status; // rax@4

  unsigned int ApiSetNameWithoutExtensionSize; // eax@5

  uintptr_t pApiSetNamePtr; // rcx@5

  __int16 ApiSetNameWithoutExtensionWordCount; // ax@8

  API_SET_NAMESPACE_ENTRY *ResolvedEntry; // rax@9

  API_SET_VALUE_ENTRY *HostLibraryEntry; // rcx@12


  ApiNamespacePtr = (__int64)ApiNamespace;
  IsResolved = 0;
  pIsResolved = Resolved;
  _ParentName = ParentName;
  Output->Length = 0;
  Output->Buffer = NULL;
  ApiSetNameBufferSize = ApiToResolve->Length;

  if (ApiToResolve->Length >= 8u)
  {
      // -------------------------

      // Check library name starts with "api-" or "ext-"

      ApiSetNameBuffer = ApiToResolve->Buffer;
      ApiSetNameBufferPrefix = ((uint64_t*) ApiSetNameBuffer)[0] &
0xFFFFFFDFFFDFFFDF;
      if (ApiSetNameBufferPrefix == API_ || ApiSetNameBufferPrefix == EXT_)
      {


        // ------------------------------

        // Compute word count of apiset library name without the dll suffix

        // E.g. :

        //     api-ms-win-core-apiquery-l1-1-0.dll -> len(api-ms-win-core-apiquery-
l1-1)

        // ------------------------------

        ApiSetNameWithoutExtensionSize = ApiSetNameBufferSize;
        pApiSetNamePtr = ((uintptr_t)ApiSetNameBuffer) + ApiSetNameBufferSize;
        do
        {
          if (ApiSetNameWithoutExtensionSize <= 1)
            break;
          ApiSetNameWithoutExtensionSize -= sizeof(wchar_t);
          pApiSetNamePtr -= sizeof(wchar_t);

        } while (((wchar_t *)pApiSetNamePtr)[0] != '-');
        ApiSetNameWithoutExtensionWordCount = (uint16_t)
ApiSetNameWithoutExtensionSize >> 1;
        // -------------------------


        if (ApiSetNameWithoutExtensionWordCount)
        {
          ResolvedEntry = ApiSetpSearchForApiSet(
            (API_SET_NAMESPACE *)ApiNamespacePtr,
            ApiSetNameBuffer,
            ApiSetNameWithoutExtensionWordCount);
          if (ResolvedEntry)
          {
            if (_ParentName && ResolvedEntry->ValueCount > 1)
            {
              HostLibraryEntry = (API_SET_VALUE_ENTRY *)ApiSetpSearchForApiSetHost(
                ResolvedEntry,
                (PWCHAR *)_ParentName->Buffer,
                _ParentName->Length >> 1,
                (API_SET_NAMESPACE *)ApiNamespacePtr);
              goto WRITING_RESOLVED_API;
            }
            if (ResolvedEntry->ValueCount > 0)
            {
              HostLibraryEntry = (API_SET_VALUE_ENTRY *)(ApiNamespacePtr +
ResolvedEntry->ValueOffset);
            WRITING_RESOLVED_API:
              IsResolved = 1;
              Output->Buffer = (wchar_t *)(ApiNamespacePtr + HostLibraryEntry-
>ValueOffset);
              Output->MaximumLength = (SHORT) HostLibraryEntry->ValueLength;
              Output->Length = (SHORT) HostLibraryEntry->ValueLength;
              goto EPILOGUE;
            }
          }
        }
      }
    }
  EPILOGUE:
    Status = STATUS_SUCCESS;
    *pIsResolved = IsResolved;
    return Status;
  }
```

Anyway I've uploaded a gist of an apiset library resolver :
https://gist.github.com/lucasg/9aa464b95b4b7344cb0cddbdb4214b25.

```
int wmain(int argc, wchar_t* argv[])
{
  if (argc < 2)
  {
    wprintf(L"ApiSetLookup : test for api set resolution.\n");
    return 0;
  }


  // Unit testing : this may not be true on your machine (that's kinda the point of
the api set schema).

  API_SET_UNIT_TEST(L"api-ms-win-crt-runtime-l1-1-0.dll", L"ucrtbase.dll");
  API_SET_UNIT_TEST(L"api-ms-win-crt-math-l1-1-0.dll", L"ucrtbase.dll");
  API_SET_UNIT_TEST(L"api-ms-win-crt-stdio-l1-1-0.dll", L"ucrtbase.dll");
  API_SET_UNIT_TEST(L"api-ms-win-core-heap-l1-1-0.dll", L"kernelbase.dll");
  API_SET_UNIT_TEST(L"api-ms-win-core-job-l1-1-0.dll", L"kernelbase.dll");
  API_SET_UNIT_TEST(L"api-ms-win-core-job-l2-1-1.dll", L"kernel32.dll");
  API_SET_UNIT_TEST(L"api-ms-win-core-registry-private-l1-1-0.dll", L"advapi32.dll");
  API_SET_UNIT_TEST(L"api-ms-win-downlevel-ole32-l1-1-1.dll", L"combase.dll");
  API_SET_UNIT_TEST(L"api-ms-win-eventing-consumer-l1-1-1.dll", L"sechost.dll");
  API_SET_UNIT_TEST(L"ext-ms-onecore-appdefaults-l1-1-0.dll",
L"windows.storage.dll");
  API_SET_UNIT_TEST(L"ext-ms-win-wer-wct-l1-1-0.dll", L"wer.dll");

  wchar_t *ApiSetLibraryName = argv[1];
  UNICODE_STRING HostApi = { 0 };
  if (ResolveApiSetLibrary(ApiSetLibraryName, &HostApi))
  {

    // HostApi.Buffer is not NULL terminated (probably to save some precious bytes
since it's COW in every process)

    wchar_t HostLibraryName[MAX_PATH];
    _snwprintf_s(HostLibraryName, _countof(HostLibraryName), HostApi.Length >> 1,
L"%s", HostApi.Buffer);


    wprintf(L"[!] Api set library resolved : %s -> %s\n", ApiSetLibraryName,
HostLibraryName);
  }
  else
  {
    wprintf(L"[x] Could not resolve Api set library : %s.\n", ApiSetLibraryName);
  }

  return 0;
}
```

That should help everyone that have a “missing” api min win import. It’s also present in my Dependencies tool:



```
MSVCP140.dll                                          AMD64   Dll; Executab
VCRUNTIME140.dll                                      AMD64   Dll; Executab
api-ms-win-crt-string-l1-1-0.dll -> ucrtbase.dll      AMD64   Dll; Executab
api-ms-win-crt-stdio-l1-1-0.dll -> ucrtbase.dll       AMD64   Dll; Executab
api-ms-win-crt-environment-l1-1-0.dll -> ucrtbase.dll AMD64   Dll; Executab
api-ms-win-crt-runtime-l1-1-0.dll -> ucrtbase.dll     AMD64   Dll; Executab
api-ms-win-crt-math-l1-1-0.dll -> ucrtbase.dll        AMD64   Dll; Executab
api-ms-win-crt-locale-l1-1-0.dll -> ucrtbase.dll      AMD64   Dll; Executab
api-ms-win-crt-heap-l1-1-0.dll -> ucrtbase.dll        AMD64   Dll; Executab
USER32.dll                                            AMD64   Dll; Executab
```

References

010 Template for parsing the ApiSet map

```
//------------------------------------------------

//--- 010 Editor v8.0 Binary Template

//

//      File:

//   Authors:

//   Version:

//   Purpose:

//  Category:

// File Mask:

//  ID Bytes:

//   History:

//------------------------------------------------


// Api namespace header

typedef struct {
  ULONG Version;
  ULONG Size <format=hex>;
  ULONG Flags;
  ULONG Count <format=hex>;
  ULONG EntryOffset <format=hex>;
  ULONG HashOffset <format=hex>;
  ULONG HashFactor <format=hex>;
} API_SET_NAMESPACE;

typedef struct {
  ULONG Hash;
  ULONG Index;
} API_SET_HASH_ENTRY;

typedef struct {
  ULONG Flags;
  ULONG NameOffset;
  ULONG NameLength;
  ULONG HashedLength;
  ULONG ValueOffset;
  ULONG ValueCount;
} API_SET_NAMESPACE_ENTRY;

typedef struct {
  ULONG Flags;
  ULONG NameOffset;
  ULONG NameLength;
  ULONG ValueOffset;
  ULONG ValueLength;
} API_SET_VALUE_ENTRY;


typedef struct {
    for (i=0; i<ApiSetMap.Count; i++) {
        API_SET_NAMESPACE_ENTRY Entry;
    }
} API_SET_NAMESPACE_ENTRIES;

typedef struct {
    for (i=0; i<ApiSetMap.Count; i++) {
        API_SET_HASH_ENTRY HashEntry;
    }
} API_SET_NAMESPACE_HASH_ENTRIES;

typedef struct (API_SET_VALUE_ENTRY &ValueEntry) {

    local int StartAddress = FTell();
    local int ValueLength = ValueEntry.ValueLength;

    CHAR Value[ValueEntry.ValueLength + 2];
} API_SET_ENTRY_VALUE;

typedef struct (API_SET_NAMESPACE_ENTRY& Entry) {
    local int ValueCount = Entry.ValueCount;
    local int NameLength = Entry.NameLength;
    local int StartAddress = FTell();
    local int HasValues = false;

    CHAR Name[Entry.NameLength + 2];

    FSeek(StartAddr + Entry.ValueOffset);

    API_SET_VALUE_ENTRY ValueEntry;
    FSeek(StartAddr + ValueEntry.ValueOffset);

    if (ValueEntry.ValueOffset)
    {
        HasValues = true;
        for (j = 0; j < Entry.ValueCount;j++)
        {
            API_SET_ENTRY_VALUE ValueName(ValueEntry);
        }
    }

    if (i < ApiSetMap.Count - 1){
        FSeek(StartAddr + Entries.Entry[i+1].NameOffset);
    }
    else {
        FSeek(StartAddr + ApiSetMap.Size);
    }

} API_SET_ENTRY_PARSED <read=ReadApiSetEntry>;

string ReadApiSetEntry(API_SET_ENTRY_PARSED & ParsedEntry)
{
    local string sApiSetName = ReadWString(ParsedEntry.StartAddress,
ParsedEntry.NameLength);

    if (ParsedEntry.HasValues) {
        local string sApiSetValue =
ReadWString(ParsedEntry.ValueName[0].StartAddress,
ParsedEntry.ValueName[0].ValueLength/2);
        return  sApiSetName + " -> " + sApiSetValue;
    }

    return sApiSetName;
}

// main entry point

LittleEndian();
Printf("Parse api set schema Begin.\n");
local int StartAddr = FTell();
local int i =0;
local int j =0;

API_SET_NAMESPACE ApiSetMap <fgcolor=cGreen>;
FSeek(StartAddr + ApiSetMap.EntryOffset);

// Enumerate API_SET_NAMESPACE_ENTRY entries

API_SET_NAMESPACE_ENTRIES Entries  <fgcolor=cPurple>;

// Traverse API_SET_NAMESPACE_ENTRY entries and retrieve

// corresponding API_SET_VALUE_ENTRY entry in order to dump

// api-set and hosts library names.

for (i=0; i<ApiSetMap.Count; i++) {

    FSeek(StartAddr + Entries.Entry[i].NameOffset);
    API_SET_ENTRY_PARSED ParsedEntry(Entries.Entry[i]) <fgcolor=cBlue>;
}

// Enumerate API_SET_HASH_ENTRY entries

FSeek(StartAddr + ApiSetMap.HashOffset);
```

```
API_SET_NAMESPACE_HASH_ENTRIES HashEntries <fgcolor=cRed>;

Printf("Parse api set schema End.\n");
```