

# Re: Address Verification

---

*Source:*

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2006-08/msg00205.1>

---

- *From:* [soviet\\_bloke@xxxxxxxxxxxx](mailto:soviet_bloke@xxxxxxxxxxxx)
  - *Date:* 23 Jul 2006 09:30:29 -0700
- 

Hi Doron

this is not a function that drivers should call b/c the context in which it is called correctly is only w/in the memory manager itself.

But how come that it is documented in DDK, and documentation does not say that this function is reserved for the system use only?????

Anton Bassov

Doron Holan [MS] wrote:

thatis not correct, i discussed this w/the owner of the memory manager. this is not a function that drivers should call b/c the context in which it is called correctly is only w/in the memory manager itself.

d

--

Please do not send e-mail directly to this alias. this alias is for newsgroup purposes only.

This posting is provided "AS IS" with no warranties, and confers no rights.

<[soviet\\_bloke@xxxxxxxxxxxx](mailto:soviet_bloke@xxxxxxxxxxxx)> wrote in message  
[news:1153626519.045621.85100@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:1153626519.045621.85100@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Hi Doron

MmIsAddressValid() does not tell you if it is a valid memory location,

## Re: Address Verification

just  
that the location could cause paging i/o or not. If you pass in  
a totally  
random address, the result is not defined. The memory  
location must be  
valid already before the call.

If you look at implementation of MmIsAddressValid(), you will see that it is EXTREMELY similar to that of MmGetPhysicalAddress(). First of all, it checks page directory. If page table, corresponding to the address in question, is not loaded in RAM (i.e P bit of its corresponding entry in the page directory is not set), MmIsAddressValid() returns FALSE straight away. If everything is OK, it checks page table. If page , corresponding to the address in question, is not loaded in RAM (i.e P bit of its corresponding entry in the page table is not set), it returns FALSE, otherwise it returns TRUE.

As it follows from the above explanation, MmIsAddressValid() just has no chance to return TRUE if the target address is invalid in the address space of the calling process.

Therefore, if it return TRUE, you can be 100% sure that the address is valid

Anton Bassov

Doron Holan [MS] wrote:

MmIsAddressValid() does not tell you if it is a valid memory location,  
just  
that the location could cause paging i/o or not. If you pass in  
a totally  
random address, the result is not defined. The memory  
location must be  
valid already before the call.

d

--

Please do not send e-mail directly to this alias. this alias is for newsgroup purposes only.

This posting is provided "AS IS" with no warranties, and confers no rights.



## Re: Address Verification

I need my kernel-mode driver to access my application's virtual RAM.

I know that I can simply have my driver perform a try/catch on the read (after using `KeStackAttachProcess()`), but this throws exceptions on invalid RAM (yes, I know that is the point in determining which RAM is valid and which isn't).

My second option is to use `ProbeForRead()`, but this again will throw exceptions on invalid RAM.

My third option is to verify the RAM from user land using `IsBadReadPtr()`, but yet again this throws exceptions.

How can I verify the RAM manually without throwing exceptions? Yes, I know that exceptions are the normal way of telling invalid RAM from good RAM, and it would be okay to use these methods under normal circumstance, but I am in a unique situation where I must not ever possibly throw an exception of any kind, at least during this operation.

I would be willing to go so

Re: Address Verification

far as to use the page table  
on 0xC0000000  
directly, just to avoid  
throwing exceptions.

Is that my last option?  
If so, where should I look to  
get started in order to  
correctly (and  
QUICKLY) use the table for  
this? Speed is also of high  
importance in  
this  
unique situation.

Thank you.