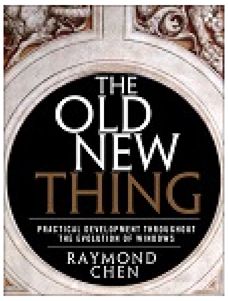
You have to tell Get- and SetSecurityInfo the object type, you can’t make it guess

devblogs.microsoft.com/oldnewthing/20250618-00/?p=111281

June 18, 2025



A customer was trying to read the security information from a process object, but their call to `GetSecurityInfo` wasn’t working.

```
PACL dacl = nullptr;
PSECURITY_DESCRIPTOR sd = nullptr;
GetSecurityInfo(processHandle,
    SE_UNKNOWN_OBJECT_TYPE,
    DACL_SECURITY_INFORMATION,
    nullptr, // not interested in owner
    nullptr, // not interested in group
    &dacl, // receives DACL
    nullptr, // not interested in SACL
    &sd); // receives security descriptor
```

They found that the call always failed with “invalid parameter”. But which parameter is invalid?

In this case, it’s the object type parameter. The `GetSecurityInfo` and `SetSecurityInfo` functions require you to tell it what kind of object for which you are trying to get or set the security information. If you pass `SE_UNKNOWN_OBJECT_TYPE`, you’re saying “I don’t know!”

Well, if you don’t know, then who does?

Process handles are kernel objects, so you should be passing `SE_KERNEL_OBJECT`.

“But why do I have to tell it what kind of object it is? Surely it can find out!”

Internally, what happens is that the different object types are managed by different providers, and the ObjectType parameter lets the function know which provider it needs to delegate the request to. I guess that in principle it could just pass the handle to every provider to say “Do you know what to do with this?”, and all of them will say “That’s not mine” except one. But that’s rather wasteful and will create tons of false alarms in program validation tools like Application Verifier.[1]

So you just have to tell it, “This is a kernel handle,” and `GetSecurityInfo` will route the request to the kernel handle provider. It’s not going to try to guess.

Even if we somehow got rid of the object type parameter from `GetSecurityInfo`, the companion function `GetNamedSecurityInfo` definitely needs an object type parameter because it needs to know how to interpret the name. For example, the string `\\alpha\beta` is ambiguous. Is this referring to the `beta` service on the computer `alpha`? Or is it referring to a printer named `beta` on that server? Or is it referring to a share named `beta` on that server? The string by itself is ambiguous,

The value `SE_UNKNOWN_OBJECT_TYPE` is a permanently invalid object type. You could use it as a sentinel value for your own `SE_OBJECT_TYPE` variables to represent an uninitialized or invalid state.

[1] And there might be multiple providers who recognize the handle. The numeric value of a service handle might happen to match the numeric value of a printer handle (say), and now the system doesn’t know whether you are asking for the security info for a service handle whose numeric value happens to be `0x1234` or for a printer handle whose numeric value happens to be `0x1234`.