MapGenericMask is just a convenience function for converting generic access to specific access, according to the instructions you provide

devblogs.microsoft.com/oldnewthing/20150515-00

May 15, 2015

Raymond Chen

For some reason, people call the `MapGenericMask` function in order to calculate what access mask to pass to request access to something. That's not what `MapGenericMask` is for. The `MapGenericMask` function is to assist the server side of the access, not the client side.

As the documentation says, the `MapGenericMask` function maps the generic access rights in an access mask to specific and standard access rights. The function applies a mapping supplied in a `GENERIC_MAPPING` structure.

This is further explained in the remarks:

> After calling the **MapGenericMask** function, the access mask pointed to by the *AccessMask* parameter has none of its generic bits (GenericRead, GenericWrite, GenericExecute, or GenericAll) or undefined bits[1] set, although it can have other bits set. If bits other than the generic bits are provided on input, this function does not clear them.

What this function does is take the `AccessMask` parameter and convert all of the `GENERIC_*` bits into object-specific bits, as defined by the `GENERIC_MAPPING`.

In other words, the code for the `MapGenericMask` function looks basically like this:

```
void MapGenericMask(
  PDWORD AccessMask,
  PGENERIC_MAPPING GenericMapping
)
{
 if (*AccessMask & GENERIC_READ)
     *AccessMask |= GenericMapping->GenericRead;

 if (*AccessMask & GENERIC_WRITE)
     *AccessMask |= GenericMapping->GenericWrite;

 if (*AccessMask & GENERIC_EXECUTE)
     *AccessMask |= GenericMapping->GenericExecute;

 if (*AccessMask & GENERIC_ALL)
     *AccessMask |= GenericMapping->GenericAll;

 *AccessMask &= ~(GENERIC_READ | GENERIC_WRITE |
                  GENERIC_WRITE | GENERIC_ALL);
}
```

The function takes the access mask and sees if any of the generic access bits are set. If so, then it replaces them with the corresponding specific access bits provided by the `GenericMapping` parameter.

Note that this function doesn't tell you anything you don't already know. It's just a helper function to make access checks easier to implement: If you are a component that manages an object and you need to perform an access check, you use `MapGenericAccess` to convert all the generic access requests into specific requests according to the rules you specified in your `GENERIC_MAPPING`, and the rest of your code only needs to deal with specific requests.

For example, we saw some time ago that a hypothetical Gizmo object could map `GENERIC_READ` to operations that query information from a Gizmo without modifying it, map `GENERIC_WRITE` to operations that alter the gizmo properties, map `GENERIC_EXECUTE` to operations that enable or disable the Gizmo, and map `GENERIC_ALL` to include all Gizmo operations.

And if you needed to do a security check on a Gizmo, you would do something like this:

```
BOOL IsGizmoAccessGranted(
 HTOKEN Token,
 PSECURITY_DESCRIPTOR SecurityDescriptor,
 DWORD AccessDesired,
 PDWORD AccessAllowed)
{
 MapGenericMask(&AccessDesired, &GizmoGenericMapping);

 BOOL AccessGranted = FALSE;
 PRIVILEGE_SET PrivilegeSet;
 DWORD PrivilegeSetSize = sizeof(PrivilegeSet);

 return AccessCheck(SecurityDescriptor,
            Token,
            AccessDesired,
            &GizmoGenericMapping,
            &PrivilegeSet,
            &PrivilegeSetSize,
            AccessAllowed,
            &AccessGranted) && AccessGranted;
}
```

When somebody wants to access a Gizmo, you use `MapGenericMask` to convert all the generic requests to specific requests. You then pass that request to `AccessCheck`, along with token for the user making the request and the security descriptor for the widget. The `AccessCheck` function does the heavy lifting of seeing which ACEs apply to the user specified by the token, then verifying that all the requested accesses have an *Allow* ACE, and that none of the requested accesses have a *Deny* ACE. It also takes care of the `MAXIMMUM_ALLOWED` access request by simply returning all the accesses that are allowed and not denied.

The point of the `MapGenericMask` function is to make life a little easier for the code that enforces security.

On the other hand, the `MapGenericMask` function is not particularly useful on the side that is requesting access. If you are requesting generic read access, just pass `GENERIC_READ`. The code that does the security check will convert the `GENERIC_READ` into the access masks that are specific to the object you are trying to access. (And it will most likely use the `MapGenericMask` function to do it.)

[1] That extra phrase "or undefined bits" is contradicted by the subsequent sentence "If bits other than the generic bits are provided on input, the function does not clear them." The second sentence is correct; the extra phrase "or undefined bits" is incorrect and should be removed.

Raymond Chen

Follow