# How to create a folder that inherits its parent’s ACL, and then overrides part of it

devblogs.microsoft.com/oldnewthing/20170223-00

February 23, 2017

Raymond Chen

A customer wants to create a folder that inherits its parent’s ACL but then overrides part of it. Specifically, the customer wanted to disallow the creation of subfolders. The customer reported that when they used the `SHCreateDirectory` function to create the folder, the folder did not inherit any ACLs at all from its parent. The only thing it got was the “deny creation of subfolders” part.

The customer provided this sample code to demonstrate what they were doing.

```
int main()
{
  PSECURITY_DESCRIPTOR pSD;
  ULONG ulSDDL;
  LPTSTR pszPath = L"C:\\my\\test\\directory";
  LPTSTR pszDacl = L"D:(D;;0x4;;;WD)";

  if (ConvertStringSecurityDescriptorToSecurityDescriptor(
    pszDacl, SDDL_REVISION_1, &pSD, &ulSDDL))
  {
    wprintf(L"Created security descriptor\n");
    SECURITY_ATTRIBUTES sa;
    sa.lpSecurityDescriptor = pSD;
    sa.nLength = sizeof(sa);
    sa.bInheritHandle = TRUE;
    if (SUCCEEDED(SHCreateDirectoryEx(nullptr, pszPath, &sa)))
    {
        wprintf(L"Created folder %s\n", pszPath);
    }
  }
  return 0;
}
```

Notice the importance of reduction, simplifying the problem to the smallest program that still demonstrates the issue. This boils the problem down to its essence, thereby allowing the development team to focus on the issue and not have to wade through (and possibly debug) unrelated code. Reduction is also a useful exercise on the part of the person reporting the problem, in order to verify that the problem really is what you think it is, rather than being a side effect of some other part of the program.

The customer added, “The ACL we are using `D:(D;;0x4;;;WD)` denies folder creation to everyone. We tried adding flags like `P`, `AI`, `OICI`, *etc.*, but none of them seem to work.”

The shell takes the security descriptor passed to the `SHCreateDirectoryEx` function and passes it through to the `CreateDirectory` function, so any issues you have with the security descriptor are really issues with the `CreateDirectory` function. The shell is just the middle man.

But even though this wasn’t really the expertise of the shell team, we were able to figure out the problem.

First off, we have a red herring: The `bInheritHandle` member controls handle inheritance, not ACL inheritance. Setting it to `TRUE` causes the handle to be inherited by child processes. But that has no effect on the ACL. And since the `CreateDirectory` function doesn’t return a handle at all, fiddling with the `bInheritHandle` means nothing since there is no handle in the first place. It’s a double red herring.

When you specify an explicit security descriptor to the `CreateDirectory` function, that establishes the security descriptor on the newly-created object. There is no inheritance from the parent. Inheritance rules are applied at creation only when you create the object with the default security attributes:[1]

> If *lpSecurityAttributes* is **NULL**, the directory gets a default security descriptor. The ACLs in the default security descriptor for a directory are inherited from its parent directory.

Passing an explicit security descriptor overrides the default behvaior.

If you want a blend of default behavior and custom behavior, then you have a few options available.

One option is to read the security descriptor of the parent object and propagate the inheritable ACEs to the child in the appropriate manner. This is a complicated endeavor and probably is best left to the experts. It’s not a simple matter of copying them from the parent to the child. You also have to to adapt the ACEs based on flags like “inherit only” and “container inherit”.

The second option is to create the directory without an explicit security descriptor and let the experts create it with the default security descriptor, which takes into account all the inheritance rules. And then modify the security descriptor post-creation to include the new ACE you want. Fortunately, MSDN has sample code for how to add an ACE to an existing security descriptor.

The customer reported that they adapted the code from MSDN and it worked perfectly.

[1] Inheritance rules are also applied when you use functions like `SetNamedSecurityInfo` and `SetSecurityInfo`.

Raymond Chen

**Follow**