

Demystifying the AdminSDHolder Object

Tony Murray

(Reprinted from WindowsItPro Magazine)

Did you ever set permissions on Active Directory (AD) objects and have those permissions mysteriously disappear shortly afterward? Most administrators involved in AD support will have experienced this odd behavior at some point. What might be causing it? Gremlins in the network? A rogue administrator? Actually, it's more likely to be the AdminSDHolder object, a little-known feature of AD designed to protect certain privileged group and user objects from compromise. We'll look at the role of AdminSDHolder and techniques you can use to modify its behavior and avoid common pitfalls.

An Example of the Problem

To illustrate, let's take a simple example of the AdminSDHolder object in action. Joe works on the IT service desk and has, via membership of the Service Desk group, the delegated permission to reset passwords for all users in an organizational unit (OU) named Wellington Users. When Joe tries to change the password for user James Smith, he receives an Access is Denied error. Being reasonably tech-savvy, Joe opens the Microsoft Management Console (MMC) Active Directory Users and Computers snap-in to compare James Smith's security settings, which Figure 1 shows, with the settings of Don Murphy, whose password he can reset, as Figure 2 shows.

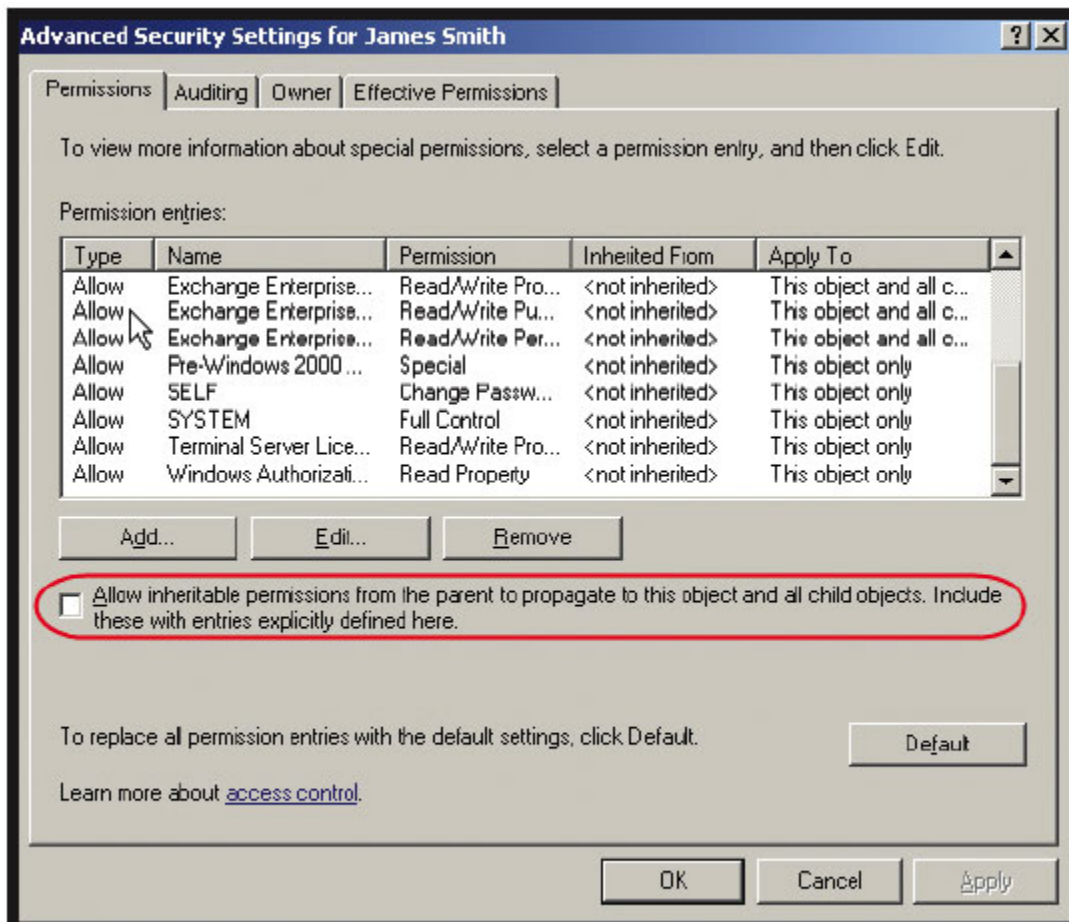
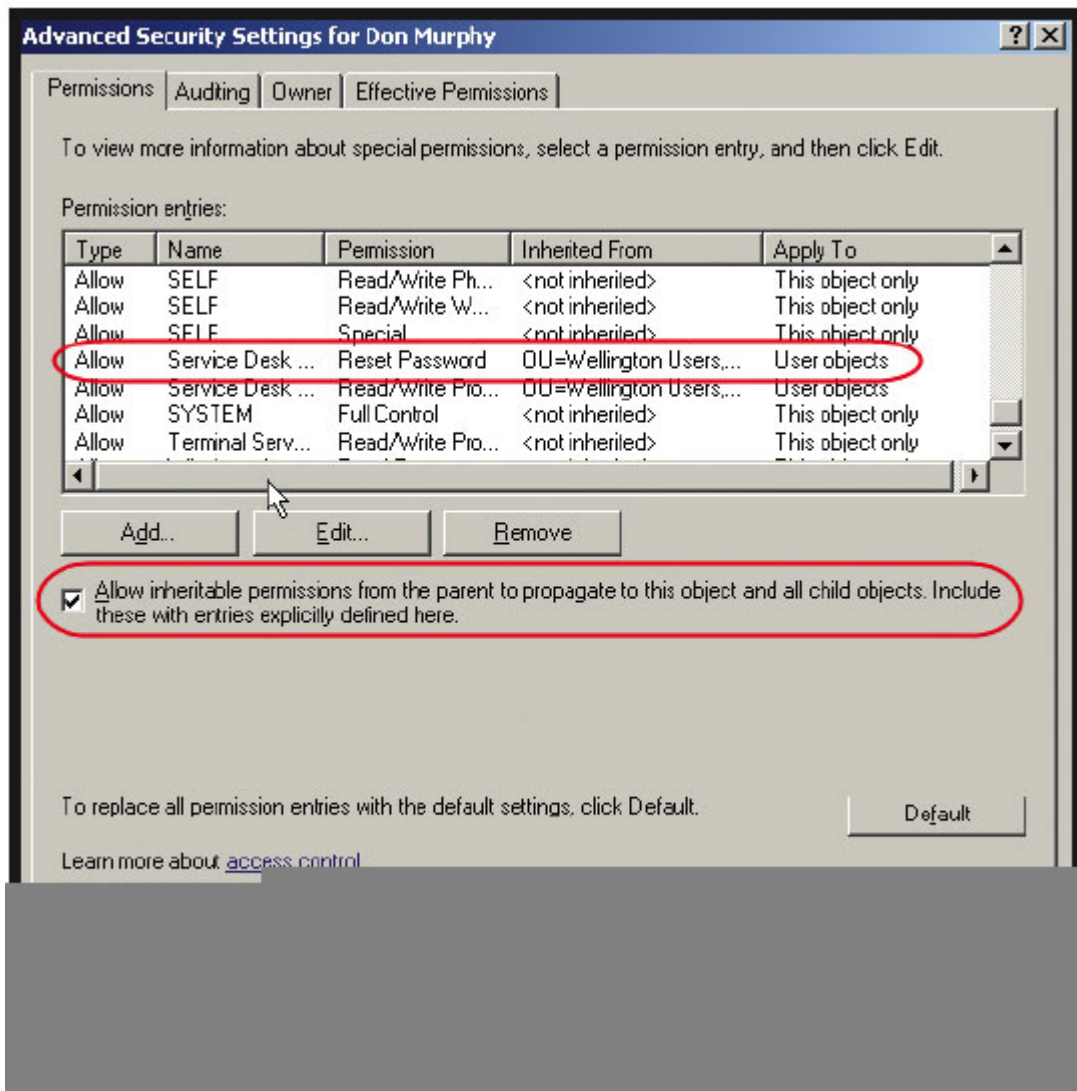


Figure 1: James Smith's AD security settings

Demystifying the AdminSDHolder Object

Tony Murray

(Reprinted from WindowsItPro Magazine)



On Don Murphy's security settings, Joe sees that the Service Desk group has permission to reset passwords. In contrast, this permission is absent from James Smith's security settings, which also show that permissions inheritance is blocked. Thinking this somewhat strange, Joe asks one of the AD administrators to change James Smith's security settings to match those of Don Murphy. Joe goes to lunch and comes back to find a note from the AD guy to say that the security settings have been changed as requested. Joe tries again to reset James Smith's password and, to his frustration, gets the same Access is Denied error. He storms off to shout at the AD admin, who tells Joe that he indeed reset James Smith's security settings. So what actually happened?

Closer inspection of James Smith's account reveals that he's a member of the Domain Admins group. AD protects certain privileged groups and accounts from being compromised, including the Domain Admins group. Table 1 lists these groups and accounts. The protection extends to all members of these groups, including nested group membership. In our example, it would be poor security for Joe to be able to reset the password for an account that's a member of Domain Admins, as he could easily then use the account himself, make his own account a member of Domain Admins, create a new account with elevated privileges, or perform similar tasks. From a security perspective, it clearly makes sense to have some type of built-in protection in place.

Demystifying the AdminSDHolder Object

Tony Murray

(Reprinted from WindowsItPro Magazine)

Table 1: Protected AD Groups and Accounts	
Type of Group	Group or Account Name
Windows 2000 Server Groups	Enterprise Admins
	Schema Admins
	Domain Admins
	Administrators
Windows Server 2003 (and Win2K Server SP4) Groups	Enterprise Admins
	Schema Admins
	Domain Admins
	Administrators
	Account Operators
	Server Operators
	Print Operators
	Backup Operators
Users	Cert Publishers
	Administrator
	Krbtgt

What's AdminSDHolder?

Many people tend use the term AdminSDHolder to refer to the process by which AD protects privileged groups and accounts, but it's really just the name of an AD object. The AdminSDHolder object is a container object in the domain directory partition at CN=AdminSDHolder, CN=System,<Domain DN>—for example, CN=AdminSDHolder, CN=System,DC=north, DC=com.

The object itself does nothing special but acts as a place-holder for a restricted security descriptor. Figure 3 shows an example of the AdminSDHolder object's security descriptor. Every hour, a background SAM task runs on the domain controller (DC) that holds the PDC emulator Flexible Single-Master Operation (FSMO) roles. This task is, among other things, responsible for checking whether the security descriptors of the protected groups and users match those of the AdminSDHolder object. If the descriptors differ, the AdminSDHolder task uses the AdminSDHolder's security descriptor to overwrite whatever permissions have been set on the problematic group or user. In our earlier example, the AdminSDHolder task replaced the inherited permissions that the AD administrator had reset on James Smith's account, which resulted in the confusion.

Demystifying the AdminSDHolder Object

Tony Murray

(Reprinted from WindowsItPro Magazine)

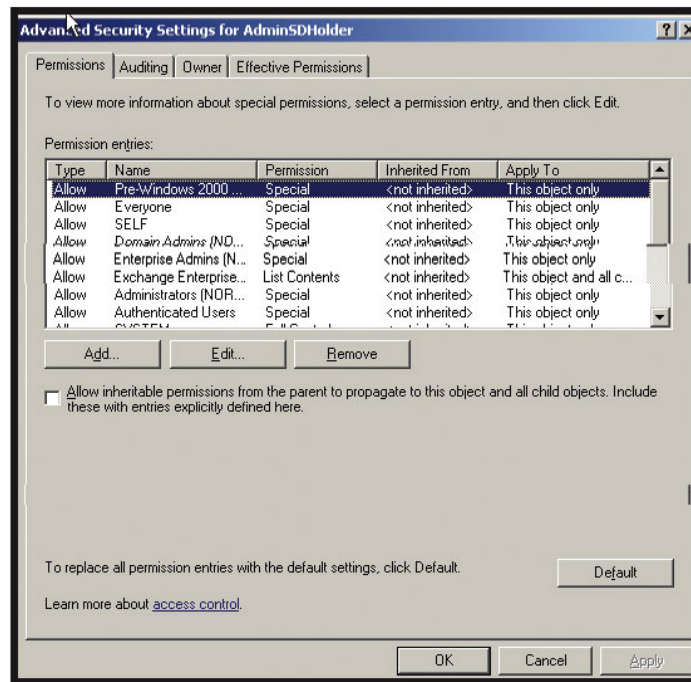


Figure 3: AdminSDHolder security descriptor

The AdminSDHolder task also sets the user's or group's adminCount attribute from <Not Set> to a value of 1. Figure 4 shows an example of James Smith's attributes viewed through the ADSI Edit tool.

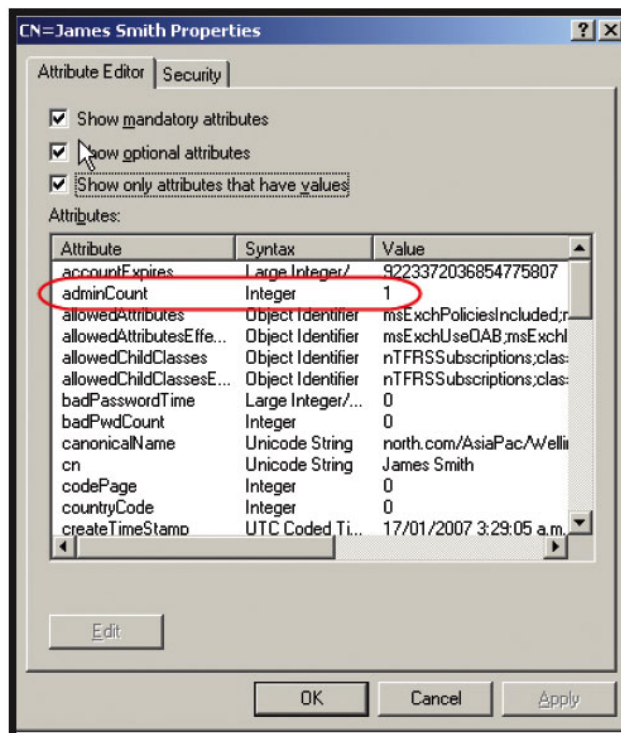


Figure 4: James Smith's attributes viewed through ADSI Edit

Demystifying the AdminSDHolder Object

Tony Murray

(Reprinted from WindowsItPro Magazine)

You can change the security descriptor associated with the AdminSDHolder object, although I don't advise doing so. Keep in mind that this change updates all protected users and groups with the new security descriptor. Don't make such a change lightly; a poorly planned configuration can leave your AD environment open to compromise. For example, assigning the Domain Users group full-control permissions on the AdminSDHolder object would effectively allow anyone in the domain to potentially elevate their group membership to Domain Admins.

Avoiding Resetting of Security Descriptors

As a best practice, I recommend that you don't make standard user accounts members of the groups protected by the AdminSDHolder object. By "standard" accounts, I mean those you typically use to log on to the domain from a workstation to perform everyday tasks, such as creating Microsoft Word documents or working with email. If a user needs to perform administrative tasks, assign the user a second account for this purpose. By doing so, you can avoid scenarios like the one I described at the beginning of the article. To remedy the problem, you could assign James Smith an additional account for administrative purposes, make his new account a member of Domain Admins, and place the account in a separate OU, to which the Service Desk group has no delegated permissions.

Changing the AdminSDHolder Task Schedule

Depending on your environment, you might decide to tighten security by running the AdminSDHolder task more frequently. You can do so by modifying the registry, as follows. (Note that the usual warnings about modifying the registry apply.)

Locate the HKEY_LOCAL_MACHINE\ SYSTEM\CurrentControlSet\Services\NTDS\ Parameters subkey.

If this subkey isn't already present, add a new DWORDvalue named AdminSDProtectFrequency and set the value (decimal) in seconds. The lowest possible setting is one minute (60 seconds), and the maximum is 120 minutes (7,200 seconds).

You should be aware that the AdminSDHolder task is resource intensive, so think carefully before you decide to modify the default setting. I've noticed that the task tends to spike the CPU on the PDC emulator DC whenever it runs. Also note that you can't force the AdminSDHolder task to run manually.

Removing Accounts from Protected Groups

In a typical AD environment, administrators regularly move accounts in and out of protected groups. For example, an account might temporarily require elevated privileges to perform a specific task. In such cases, the AdminSDHolder task applies the security descriptor associated with the AdminSDHolder object (as described earlier)—but doesn't revert to the previous security descriptor when the account is removed from the protected group. Additionally, the AdminSDHolder task doesn't remove the adminCount attribute value of 1. In my opinion, this behavior is less than ideal because it leaves behind a security descriptor that has permissions inheritance disabled and an adminCount attribute value that's potentially confusing if used for LDAP queries. I'd prefer it if the AdminSDHolder task could tidy up after itself, but alas, this isn't the case in the current version of Windows.

Determining Which Objects the AdminSDHolder Task Has Modified

The most common effective method to determine which accounts and groups have been touched by the AdminSDHolder task is to look for Event ID 684 account-management audit entries in the Security event log on the PDC emulator DC. Figure 5 shows an example of such a log entry. Note that you must have account-management auditing enabled for the system to log the required events. (For

Demystifying the AdminSDHolder Object

Tony Murray

(Reprinted from WindowsItPro Magazine)

more information about enabling AD auditing, see "Monitoring AD Changes," September 2003, InstantDoc ID 39769.)

Figure 5:

Sample Security event log entry showing Event ID 684

```
Event Type:          Success Audit
Event Source:        Security
Event Category:      Account Management
Event ID: 684
Date:                19/01/2007
Time:                5:32:03 p.m.
User:                NT AUTHORITY\ANONYMOUS LOGON
Computer: DCN1
Description:
Set ACLs of members in administrators groups:
  Target Account Name:    JamesS
  Target Domain:          DC=north,DC=com
  Target Account ID:      NORTH\JamesS
  Caller User Name:       DCN1$
  Caller Domain:          NORTH
  Caller Logon ID:        (0x0,0x3E7)
Privileges:            -
```

As an alternative method, you can log security descriptor propagation (SDPROP) activity. The AdminSDHolder task and SDPROP are unrelated, but whenever a Security Descriptor (SD) update occurs, SDPROP attempts to propagate the changes to child objects. In other words, SDPROP informational events provide clues as to what objects have been touched by the AdminSDHolder task, but be aware that SDPROP potentially logs other, unrelated activity.

SDPROP doesn't log informational events in the Directory Service (DS) event log unless you increase the level of diagnostics logging from the default. To enable SDPROP informational events, again on the PDC emulator DC, set the value of 9 Internal Processing on the following registry subkey to 5 (decimal): HKEY_LOCAL_MACHINE\SYSTEM\CurrentControl Set\Services\NTDS\Diagnostics. After you've enabled the increased logging level, look for Event IDs 1257 and 1258 in the DS event log with NTDS SDPROP as the source. The events will contain the name of the objects updated by SDPROP. Note that increasing the logging level generates a large number of events, so you should do so only to troubleshoot a particular problem, then decrease the logging level when you're finished troubleshooting.

A third method is to issue an LDAP query for all objects within the domain that have an adminCount attribute value of 1. However, this method is unreliable for the reasons I've already mentioned—that is, the fact that the AdminSDHolder task doesn't remove the attribute value when the object is no longer a member of one of the protected groups. In the following example, I use ADFind (an excellent freeware command-line tool from joeware .net, <http://www.joeware.net>) to run the LDAP query. The query searches the default directory partition (the domain partition) and returns all user objects that have an adminCount attribute value of 1. (The query wraps to multiple lines because of space constraints, but you should type it on one line.)

Demystifying the AdminSDHolder Object

Tony Murray

(Reprinted from WindowsItPro Magazine)

```
adfind -default -f "(&(objectClass=user)  
  (objectCategory=Person)(adminCount=1))"
```

Avoid the Pitfalls

The AdminSDHolder task leverages the security descriptor associated with the AdminSDHolder container object to protect certain AD objects from compromise. Although this feature is necessary from a security perspective, it can cause confusion for administrators in certain delegation scenarios. Understanding AdminSDHolder's behavior can help administrators avoid the pitfalls and also help with troubleshooting. You can also preempt common mistakes by following delegation best practices and implementing secondary accounts for administrative purposes.