

Windows Server 2012: Group Managed Service Accounts

Remember when Windows Server 2008 R2 was released, and one of the exciting new features was [Managed Service Accounts](#)? Managed Service Accounts (MSAs) held so much promise – automatic password management and automatic SPN registration. [Remember all of those service you have in the domain, that are over-privileged, and whose passwords haven't changed in the past 5 years?](#) You dreamed of replacing them with MSAs, and then you read the fine-print. MSA's are not supported for applications like Exchange or SQL. MSA's cannot be shared across multiple hosts. MSA's cannot even be used to run a scheduled task. After we totally harshed your mellow, you didn't even bother to check with your 3rd party vendors to see if their applications could use MSAs.

Enter Windows Server 2012 Group Managed Service Accounts

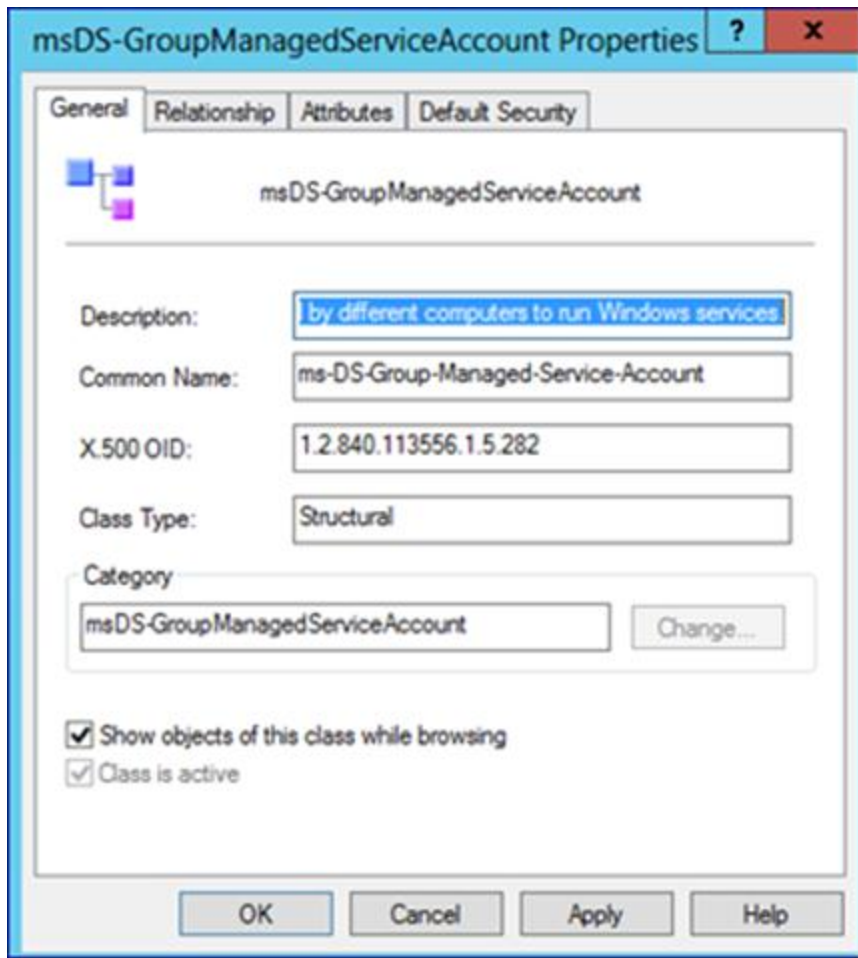
Windows Server 2012 has come to the rescue with the Group Managed Service Account (gMSA). Think of Group Managed Service Accounts as a usable version of the Managed Service Account. With gMSAs, Windows Server 2012 has addressed most of the limitations of MSAs. Specifically:

- A single gMSA can be used on multiple hosts.
- A gMSA can be used for scheduled tasks.
- A gMSA can be used for IIS Application Pools, SQL 2012 and potentially other applications - check with the vendor :)

Now is the time to learn about Group Managed Service Account, and test their potential use in your environment.

What is a gMSA?

When you extend your schema for Windows Server 2012, a new object class is added for gMSAs – msDSGroupManagedServiceAccount.



It's derived from the computer class, with five new/additional attributes:

- msDS-GroupMSAMembership – Governs which computers (groups of computers) are allowed to retrieve the password and make use of the gMSA.
- msDS-ManagedPassword – a binary blob containing (among other things) the current password, previous password and password change interval.
- msDS-ManagedPasswordInterval – configured at account creation (can't be changed later), determines how often (number of days) the password must be changed.
- msDS-ManagedPasswordID – key identifier used by Key Distribution Service (KDS) to generate current password.
- msDS-ManagedPasswordPreviousID – key identifier from previous password.

Password Behavior

Unlike the previous MSAs, the password for gMSAs are generated and maintained by the Key Distribution Service (KDS) on Windows Server 2012 DCs. This allows multiple hosts to use the gMSA. Member servers that wish to use the gMSA, simply query the DC for the current password. Usage of the gMSA is restricted to only those computers specified in the security descriptor, msDS-GroupMSAMembership.

As the password for the gMSA is needed, for example when a host using the gMSA retrieves it, the DC will determine if a password change is necessary. If so, it uses a pre-determined algorithm to compute the password (120 characters). This algorithm depends upon a root key ID that is shared across all Windows Server 2012 KDS instances (pre-generated by an administrator), the current time (translated to an epoch) and the SID of the gMSA. Thus, any Windows Server 2012 KDS can generate the password, and all KDS instances use the same algorithm and will generate the same password.

Since the password (or, more precisely, the password hash) for the gMSA will be stored in Active Directory, down-level DCs will still be able to handle authentication requests – for example, to respond to a Kerberos TGS-REQ for a service ticket.

Group Managed Service Accounts Requirements

- At least one Windows Server 2012 Domain Controller
- A Windows Server 2012 or Windows 8 machine with the ActiveDirectory PowerShell module, to create/manage the gMSA.
- A Windows Server 2012 or Windows 8 domain member to run/use the gMSA.

Using Group Managed Service Accounts

Like most new features in Windows Server 2012, creating/configuring gMSAs are easy. In essence, there are three steps:

1. Create the KDS Root Key (only has to be done once per domain).
2. Create and Configure the gMSA
3. Configure the gMSA on the host(s)

Let me demonstrate with an example.

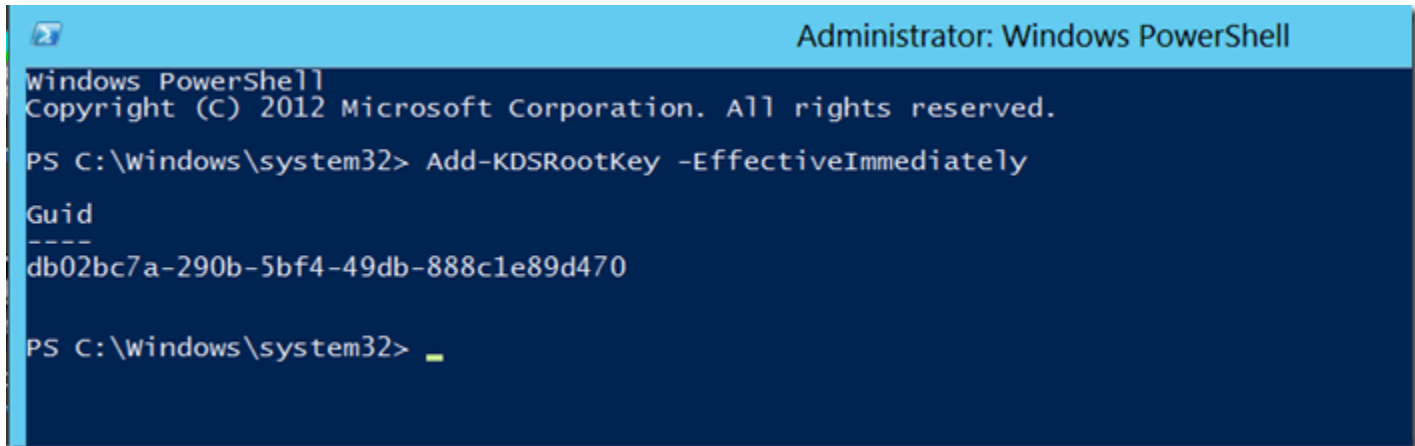
Using a gMSA for a Scheduled Task

I've got customers that run scheduled tasks on domain controllers. For example, they like to run a batch file nightly to perform a custom job. Unfortunately, some of these custom jobs require that the account that runs the process be an Administrator, or Domain Admin. Since they schedule tasks across multiple domain controllers, using the same account, they rarely (if ever) change the password. As soon as they upgrade their DCs to Server 2012, I'll be nagging them to transition to a gMSA. Here's how:

1. Create the KDS Root Key (only once per domain). This is used by the KDS service on DCs (along with other information) to generate passwords.

From a Windows Server 2012 Domain Controller (or Windows Server 2012/Windows 8 host with the ActiveDirectory PowerShell module) run:

```
Add-KDSRootKey -EffectiveImmediately
```



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2012 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Add-KDSRootKey -EffectiveImmediately

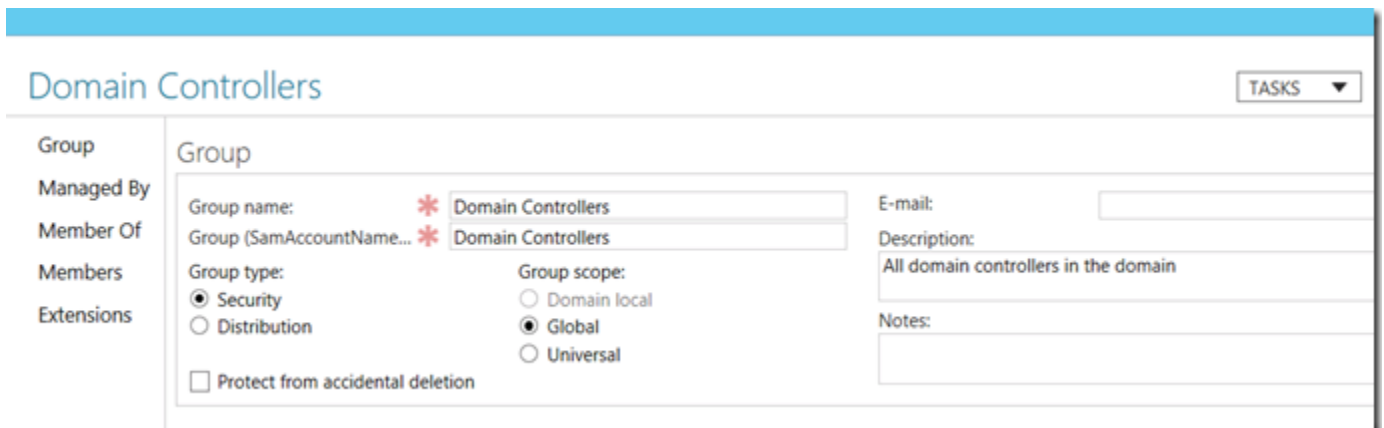
Guid
----
db02bc7a-290b-5bf4-49db-888c1e89d470

PS C:\Windows\system32> _
```

Then go home and continue tomorrow 😊. Seriously, in spite of what you might think `-EffectiveImmediately` means wait up to 10 hours. This is a safety measure to make sure all DCs have replicated and are able to respond to gMSA requests. There is [a trick to bypass this safety measure](#), but should only be used in a lab.

2. Create and configure the gMSA

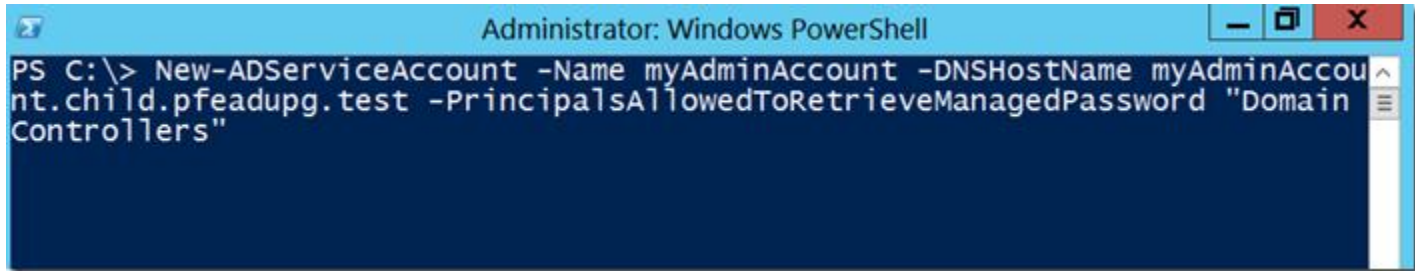
First, identify or create a security group and add the computer objects of the hosts that will be allowed to use the gMSA. While you could grant individual computer objects the ability to use the gMSA, creating a security group to hold these computer objects will give you more administrative flexibility going forward. The only downside to using a group, is that computers/hosts will need to be re-booted after being added/removed from the group to reflect membership changes. In this example, the hosts are domain controllers and there is already a group for Domain Controllers (which contains all the DCs in the domain) that I can leverage:



Next, you must use PowerShell (with the Server 2012 AD cmdlets) to create the gMSA. During the creation you must specify a name (SamAccountName) and dnsname. You'll also want to specify the group allowed to use the gMSA (see above) and potentially SPNs for the account:

```
New-ADServiceAccount -name <ServiceAccountName> -DNSHostName <fqdn> -
PrincipalsAllowedToRetrieveManagedPassword <group> -ServicePrincipalNames
<SPN1,SPN2,...>
```

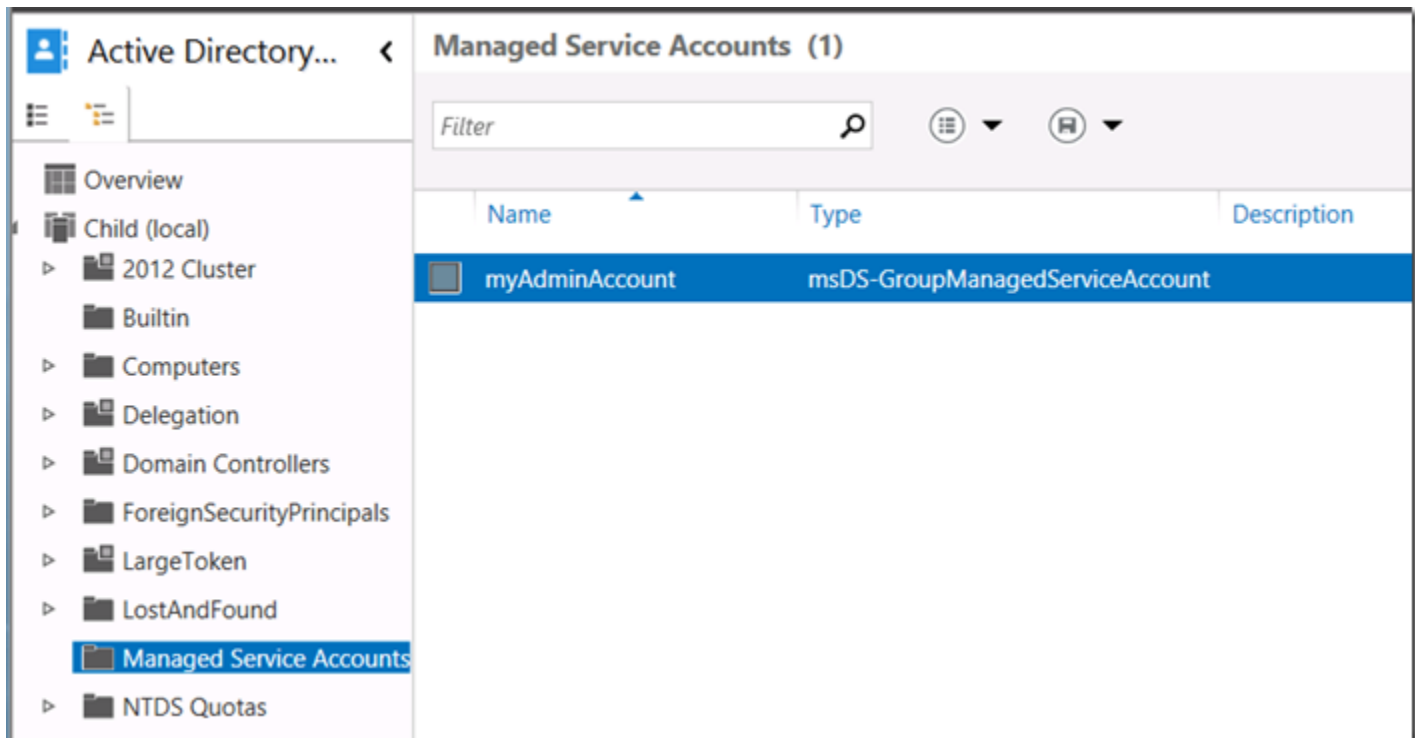
In my case, I'll only specify the Name, DNSHostName, and PrincipalsAllowedToRetrieveManagedPassword:



```
Administrator: Windows PowerShell
PS C:\> New-ADServiceAccount -Name myAdminAccount -DNSHostName myAdminAccount.child.pfeadupg.test -PrincipalsAllowedToRetrieveManagedPassword "Domain Controllers"
```

If you get a “key does not exist” error you forgot to do Step 1, or you were too impatient.

If everything works as expected, you'll notice a new gMSA object in your domain's Managed Service Accounts OU:



3. Configure the gMSA on the host

First, you'll want to install and test the gMSA on the host. While this isn't always necessary, it's safe practice. You'll run the following PowerShell cmdlets on the host which will be using the gMSA:

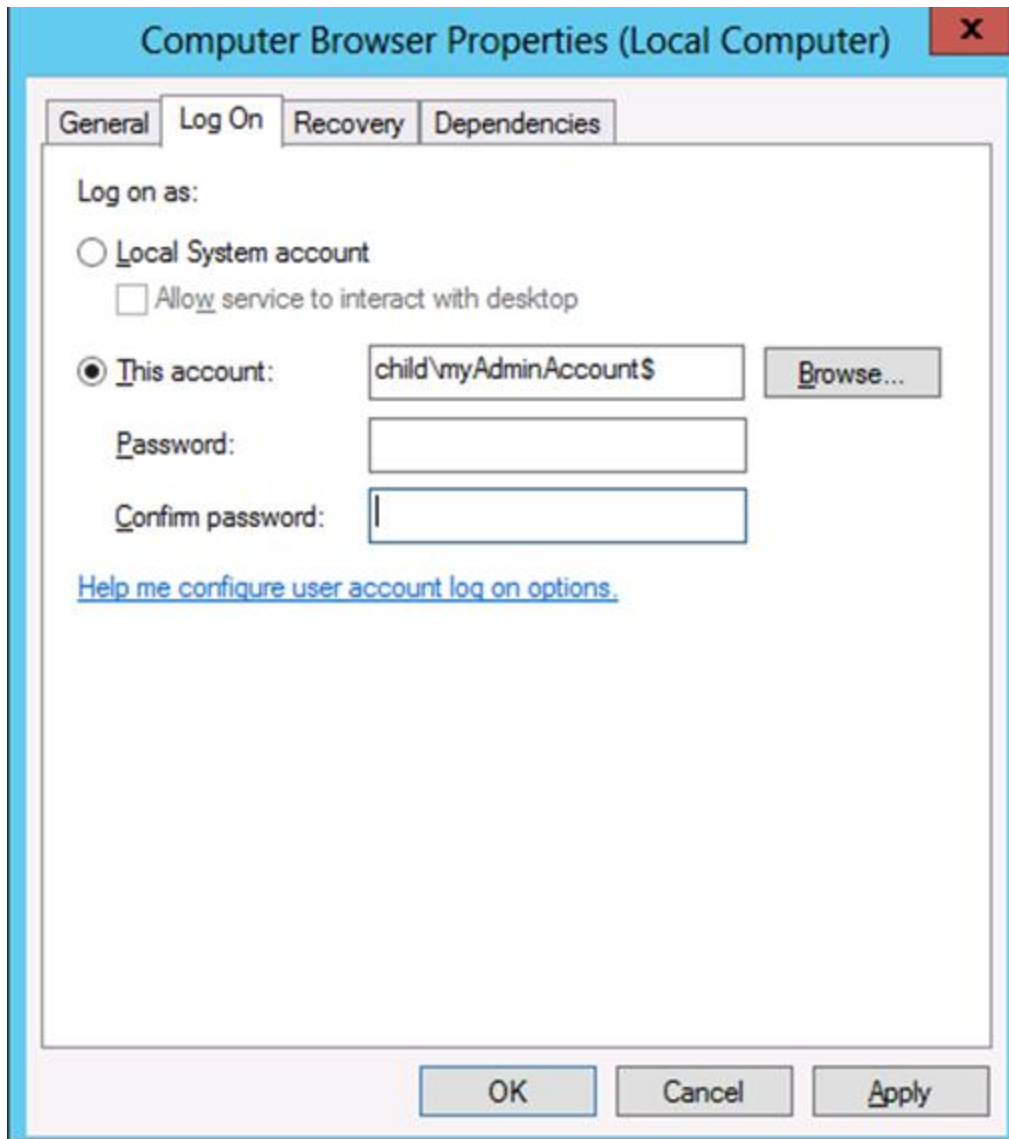
```
Install-AdServiceAccount <gMSA>
```

```
Test-AdServiceAccount <gMSA>
```

```
Administrator: Windows PowerShell
PS C:\> Install-ADServiceAccount myAdminAccount
PS C:\> Test-ADServiceAccount myAdminAccount
True
PS C:\>
```

You'll notice the Test-ADServiceAccount, returns "True". If it returns "False", it should include a verbose error message.

Next, if you are going to use the gMSA for a Service, an IIS Application Pool, or SQL 2012, you would simply plug it in the Logon/Credentials UI. The trick is to append a \$ after the account name, and leave the password blank:



If necessary, Windows will grant the account the "Log On As a Service" right, and once the service is started, the password will be automatically retrieved from a Windows Server 2012 DC.

4. Using the gMSA for a Scheduled Task

Since my use case was a scheduled task, I'll show you some of the interesting nuances around gMSA and scheduled tasks. The fundamental problem is that you can't use the Task Scheduler UI. So we'll use PowerShell cmdlets, instead. (You could also use `schtasks.exe` with an XML config file, but I'll let you figure that one out yourself).

To use the PowerShell cmdlets, you need to define an Action (what), a Trigger(when) and a Principal(under which identity):

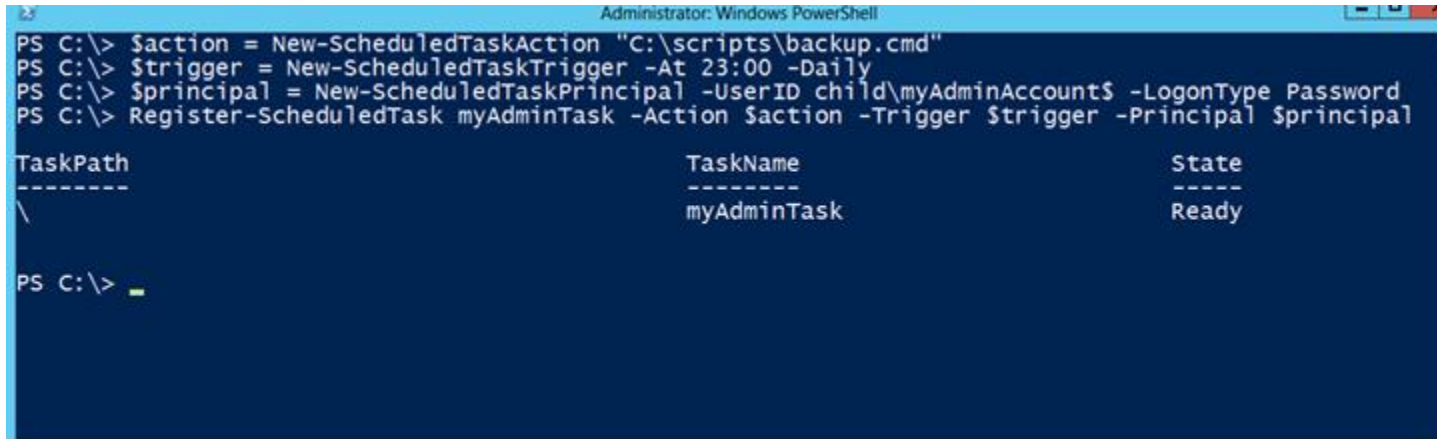
```
$action = New-ScheduledTaskAction "c:\scripts\backup.cmd"  
$trigger = New-ScheduledTaskTrigger -At 23:00 -Daily
```

```
$principal = New-ScheduledTaskPrincipal -UserID child\myAdminAccount$ -LogonType Password
```

After the `-LogonType` switch you type the word `Password`, and not an actual password. This tells the scheduled task to retrieve the actual password for the gMSA from a domain controller.

Now you plug these three variables into a `Register-ScheduledTask` cmdlet

```
Register-ScheduledTask myAdminTask -Action $action -Trigger $trigger -Principal $principal
```

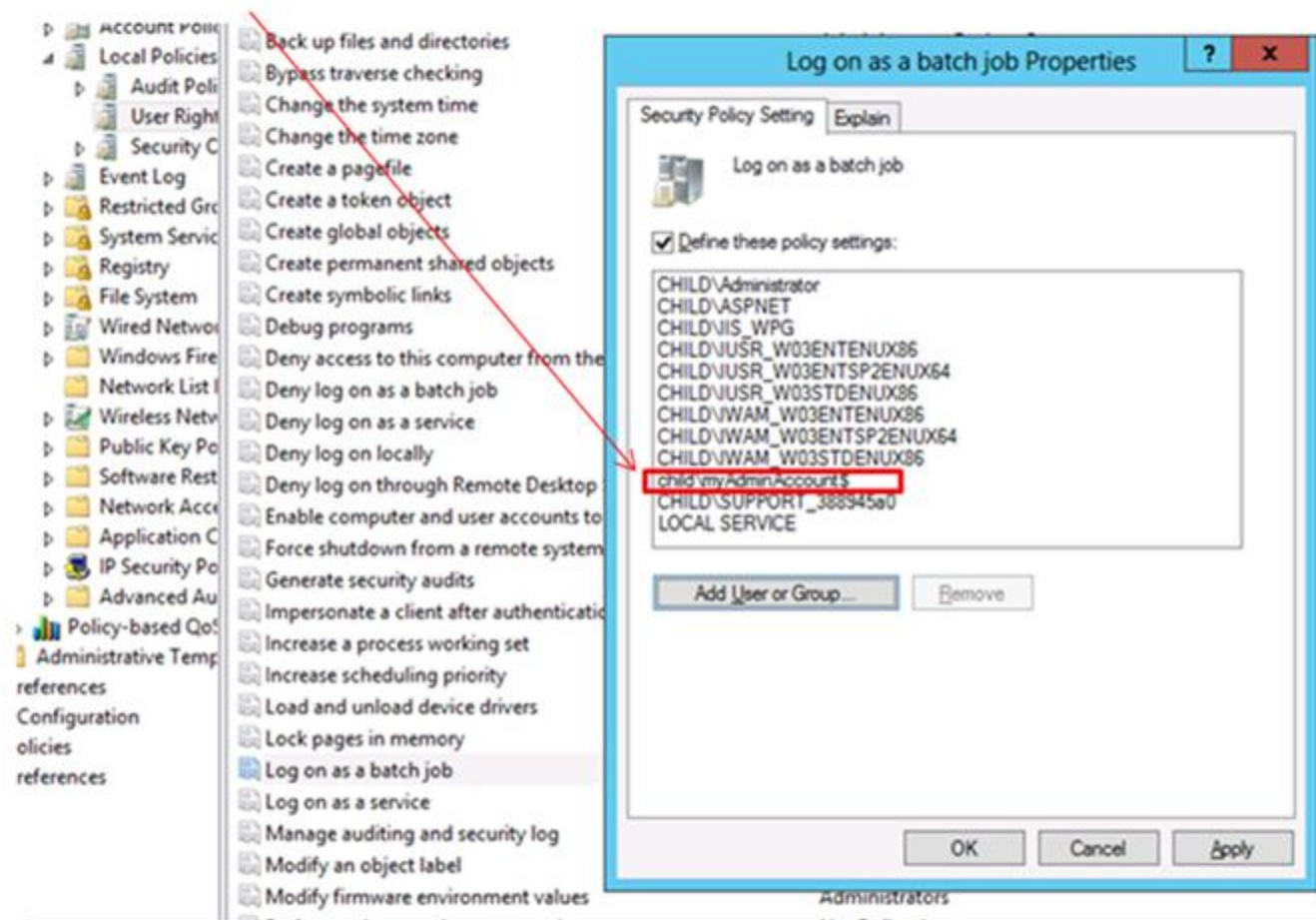


```
Administrator: Windows PowerShell
PS C:\> $action = New-ScheduledTaskAction "C:\scripts\backup.cmd"
PS C:\> $trigger = New-ScheduledTaskTrigger -At 23:00 -Daily
PS C:\> $principal = New-ScheduledTaskPrincipal -UserID child\myAdminAccount$ -LogonType Password
PS C:\> Register-ScheduledTask myAdminTask -Action $action -Trigger $trigger -Principal $principal

TaskPath          TaskName          State
-----
\                  myAdminTask       Ready

PS C:\> _
```

Be aware: If you are using the gMSA to run scheduled batch jobs/scripts, you will have to grant the gMSA the ability to "Log on as a batch job" on the machine:



You may also need to grant the gMSA membership in a local group (like Administrators, or Backup Operators) so it has the necessary rights to accomplish the task.

Open the Task Scheduler and you should see your scheduled task listed. You should be able to manually execute the task (to test it). Otherwise, you cannot use the gui to edit the task. Changes have to be made using PowerShell cmdlets.

