Retrieve SSH public key from Active Directory for SSH authentication

Sysac the N saotn.org/windows-server/retrieve-ssh-public-key-from-active-directory-for-ssh-authentication

Jan Reilink October 29, 2021

If you want to be able to log on to your Windows Servers through Win32-OpenSSH, you can make use of key-based authentication in OpenSSH through a `~/.ssh/authorized_keys` file. But if you have tens (hundreds) of servers and/or users, perhaps it's easier to retrieve user SSH public keys from Active Directory (AD). In this article I'll explain how.

If you have tens (hundreds) of servers and/or users it's easier to retrieve user SSH public keys from Active Directory (AD) than from various unmanaged `authorized_keys` files. This post outlines the steps necessary.

Configure Active Directory (AD) Schema for SSH publick key authentication

How to configure SSH public key authentication for Windows Server in Active Directory (AD)

To configure SSH public key-based authentication in OpenSSH for Windows Server, you first you need to extend your AD schema to allow for the storage of public keys. Just follow the great steps Ted Salmon posted at Storing SSH keys in Active Directory for easy deployment. It doesn't really matter if you name your attribute `sshPublicKeysshPublicKeys`.

For the sake of this post, and to have all steps complete, I've redone the steps here.

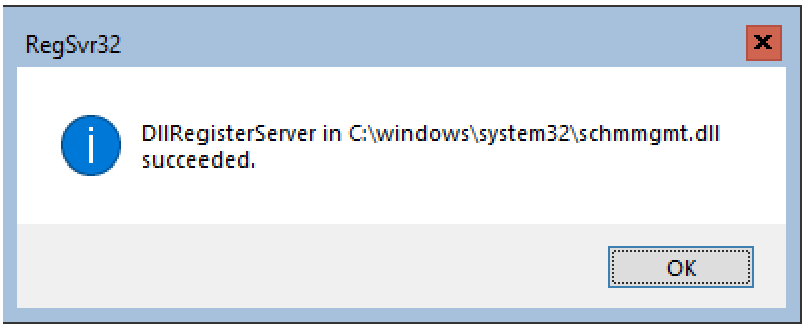
Extend the Active Directory Schema

First, as an administrator – I'm sure you are – start your PowerShell prompt, and execute the following command to create a new DWORD key `Schema Update Allowed1`:

```
# set Schema Update Allowed to true / 1
#
New-Itemproperty "HKLM:\SYSTEM\CurrentControlSet\Services\NTDS\Parameters" `
  -Name "Schema Update Allowed" `
  -Value "1" `
  -PropertyType DWORD `
  -Force
```

When in doubt you can always use the `regedit.exe.reg` file and silently import .reg file in your Windows registry. Otherwise the schema cannot be updated. Do this on the domain controller having the Schema master FSMO rolenetdom query fsmo).

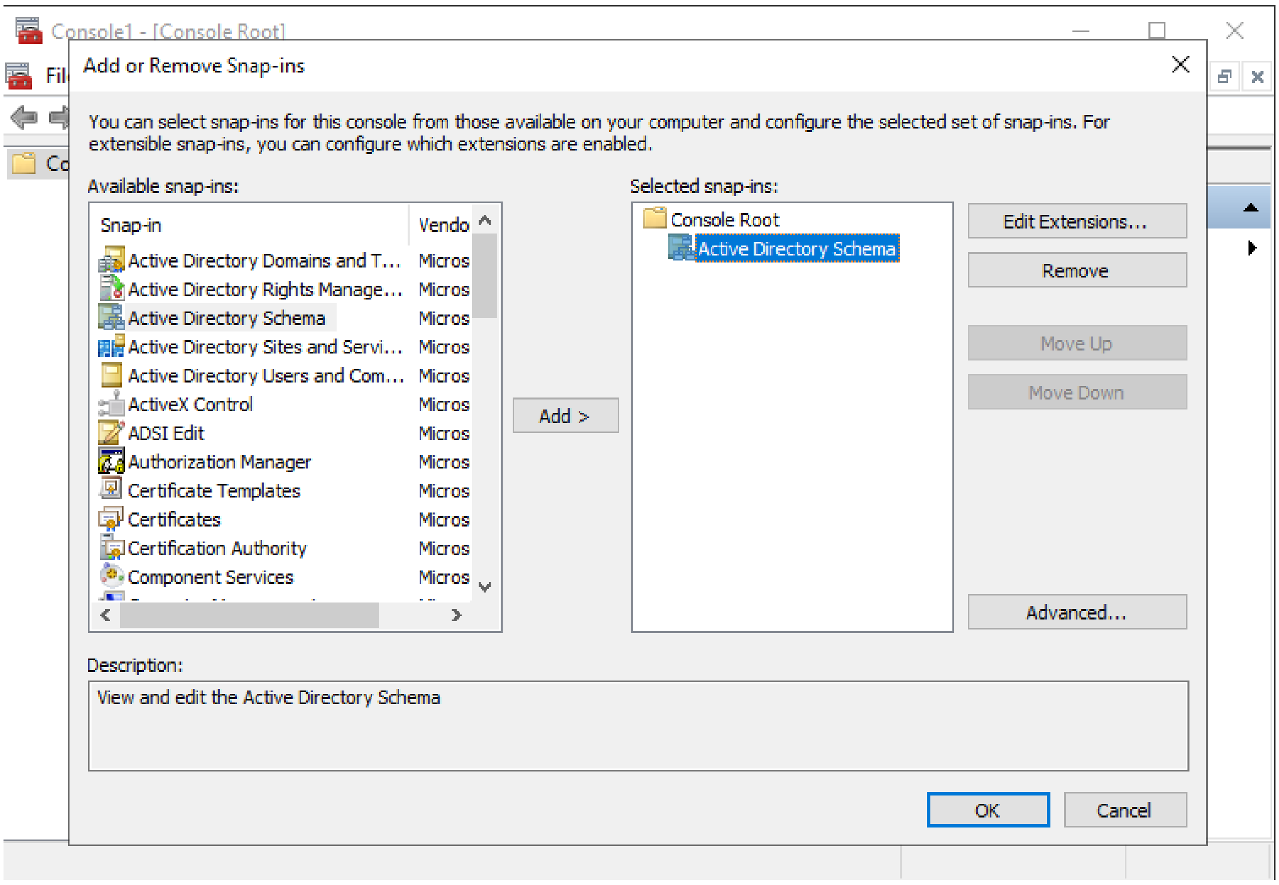
Secondly, run `regsvr32.exe C:\windows\system32\schmmgmt.dll` to enable the Schema Editor snap-in for mmc.


RegSvr32

DllRegisterServer in C:\windows\system32\schmmgmt.dll succeeded.

enable the Schema Editor snap-in for mmc

Now if you run `mmc`


Console1 - [Console Root]

Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendo
Active Directory Domains and T...	Micros
Active Directory Rights Manage...	Micros
Active Directory Schema	Micros
Active Directory Sites and Servi...	Micros
Active Directory Users and Com...	Micros
ActiveX Control	Micros
ADSI Edit	Micros
Authorization Manager	Micros
Certificate Templates	Micros
Certificates	Micros
Certification Authority	Micros
Component Services	Micros

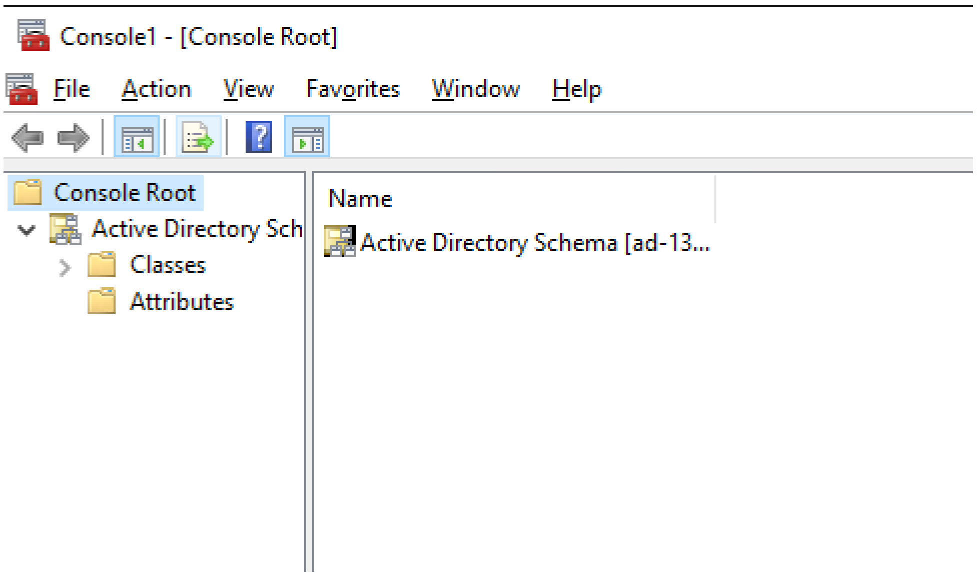
Selected snap-ins:

Console Root
Active Directory Schema

Description:

View and edit the Active Directory Schema

Select Active Directory Schema snap-in for MMC


Console1 - [Console Root]
File Action View Favorites Window Help
Console Root
Active Directory Sch
Classes
Attributes
Name
Active Directory Schema [ad-13...

Active Directory Schema, Classes and Attributes in AD

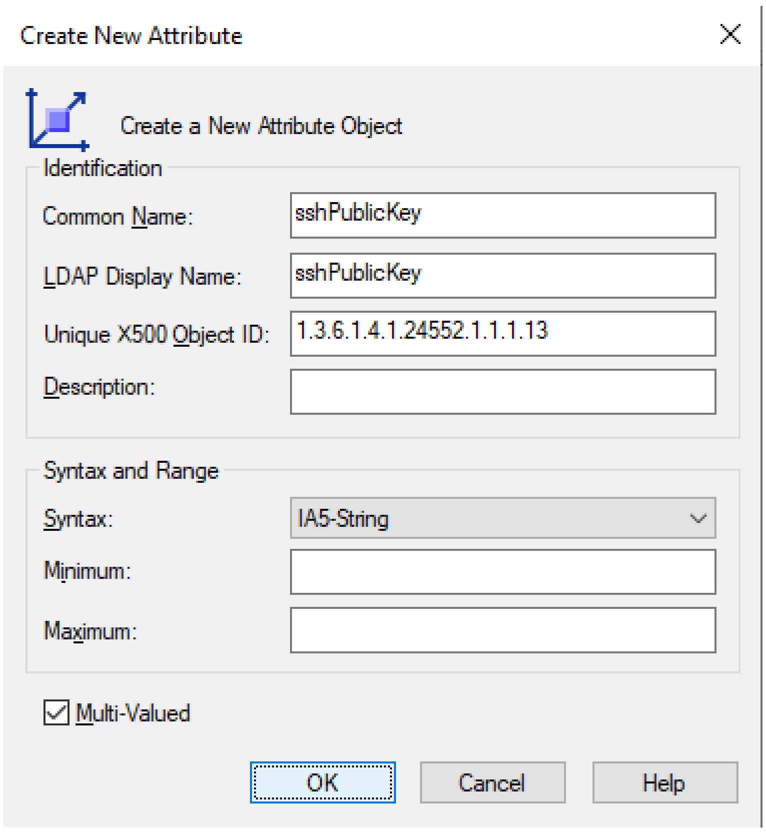
This is what you need to extend the Active Directory Schema.

The following steps might feel a bit tensive. Just keep calm. If you're not already, make sure you are an *Schema Admin*.

Add a new Attribute

You first need to create a new Attribute in your AD Schema. Follow these steps:

1. Right click Attributes and click Create New Attribute. Click Continue if a warning titled "Schema Object Creation" appears.
2. Create a New Attribute Object values:
    - Common Name: `sshPublicKey`
    - LDAP Display Name: `sshPublicKey`
    - Unique X500 Object ID: `1.3.6.1.4.1.24552.1.1.1.13`
    - Syntax: select `IA5-String`
    - Minimum: can be left blank
    - Maximum: can be left blank
    - select Multi-Valued check box


Create New Attribute

Create a New Attribute Object

Identification

Common Name: sshPublicKey

LDAP Display Name: sshPublicKey

Unique X500 Object ID: 1.3.6.1.4.1.24552.1.1.1.13

Description:

Syntax and Range

Syntax: IA5-String

Minimum:

Maximum:

[✓] Multi-Valued

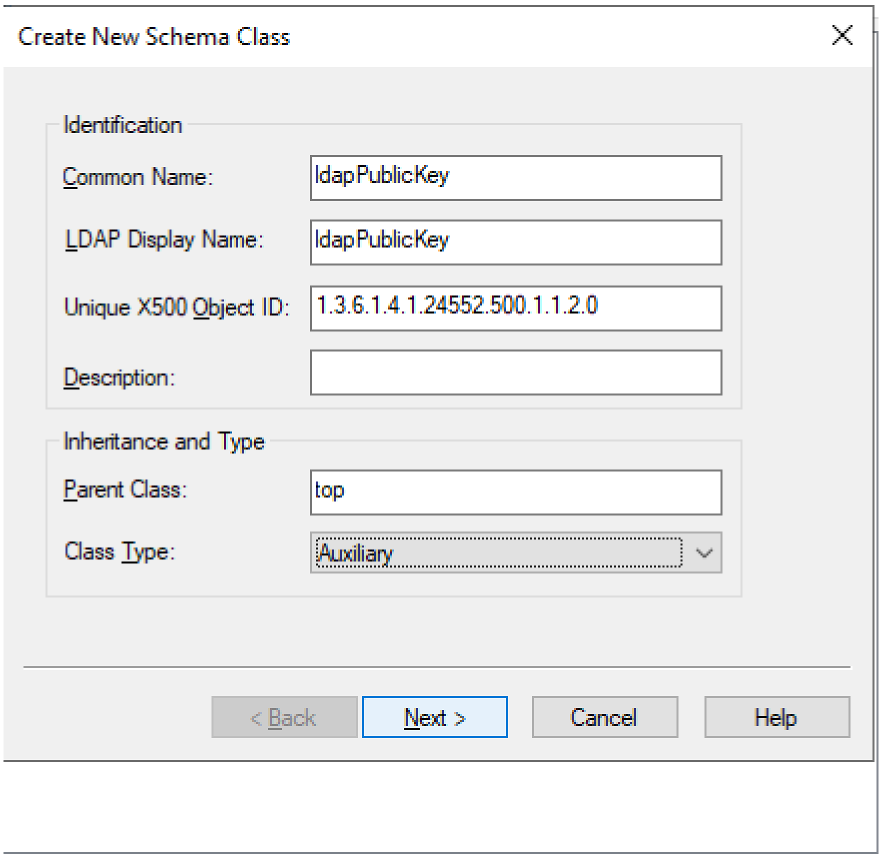
Create New sshPublicKey Attribute

Now you can create a class for the attribute.

Add a new Class

As with the schema attribute, follow the steps to create a new class.

1. Right click Classes and select Create Class
   Click Continue if a warning titled “Schema Object Creation” appears.
2. Create New Schema Class values:
   - Common Name: `ldapPublicKey`
   - LDAP Display Name: `ldapPublicKey`
   - Unique X500 Object ID: `1.3.6.1.4.1.24552.500.1.1.2.0`
   - Parent Class: `top`
   - Class Type: select `Auxiliary`


Create New Schema Class

Identification

Common Name: ldapPublicKey

LDAP Display Name: ldapPublicKey

Unique X500 Object ID: 1.3.6.1.4.1.24552.500.1.1.2.0

Description:

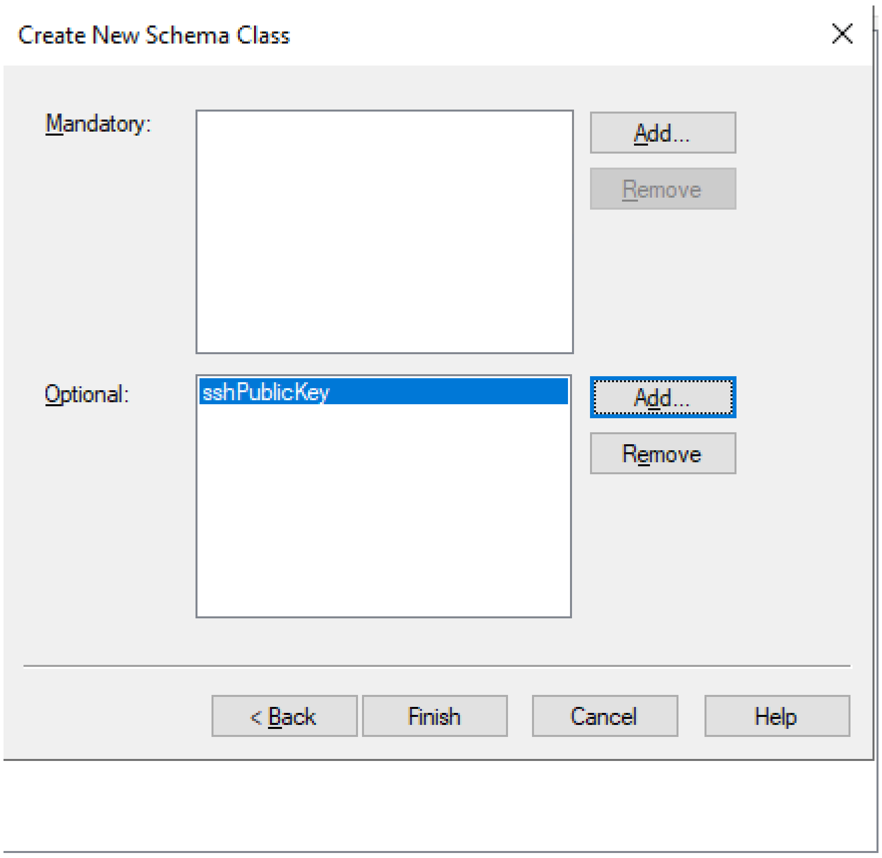
Inheritance and Type

Parent Class: top

Class Type: Auxiliary

new Schema Class ldapPublicKey

3. Click Next
4. Under Optional Add and Select `sshPublicKey`


Create New Schema Class

Mandatory:

Optional: sshPublicKey

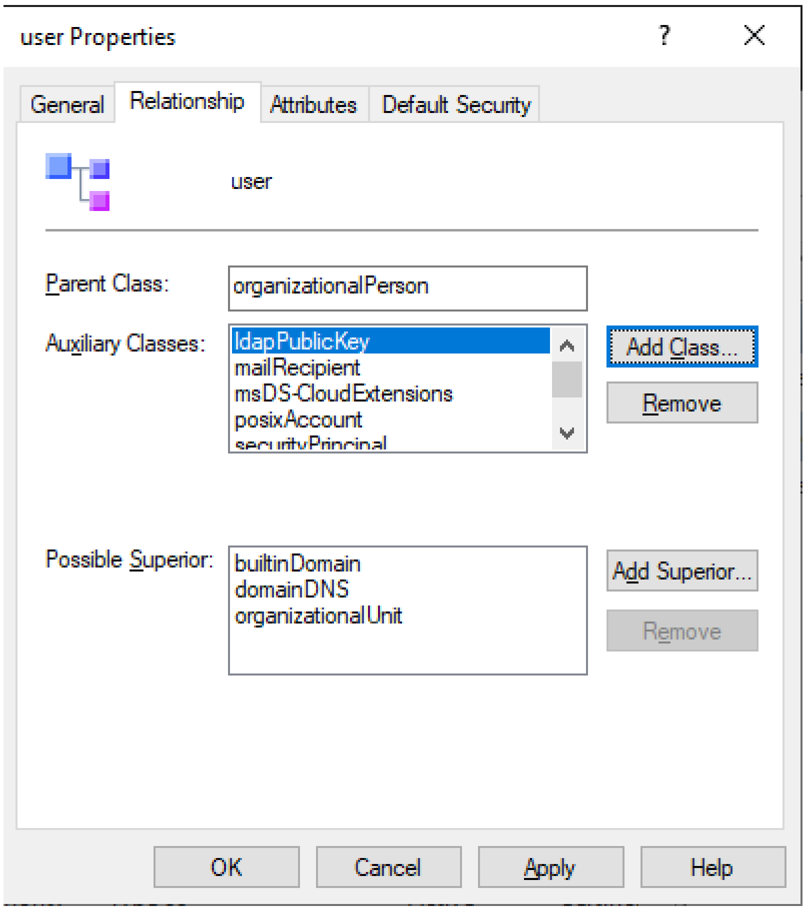
Add Attribute sshPublicKey

5. Click Finish

Associate class to user objects

The class `ldapPublicKeysshPublicKey`

1. Expand Classes, right click user and select Properties
2. Click on the Relationship tab
   - click Add Class under Auxiliary Classes
   - select `ldapPublicKey`, click OK
   - click Apply



Parent Class: organizationalPerson

Auxiliary Classes: ldapPublicKey, mailRecipient, msDS-CloudExtensions, posixAccount, securityPrincipal

Possible Superior: builtinDomain, domainDNS, organizationalUnit

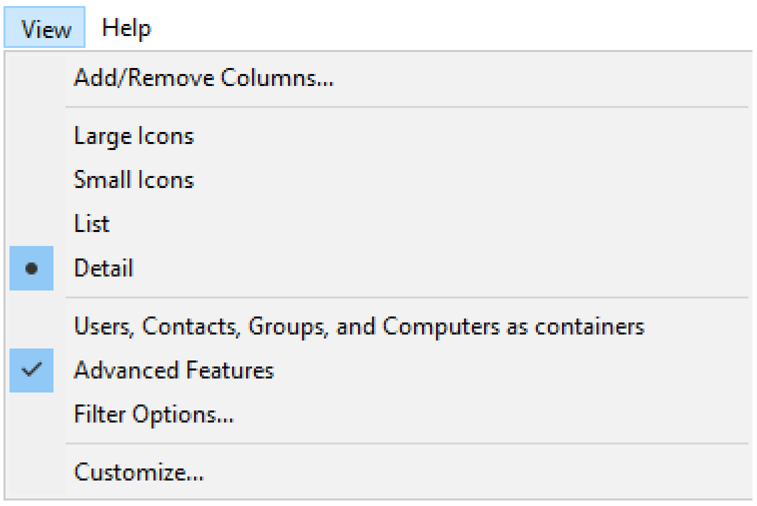
Associate ldapPublicKey with a user object

3. click OK to close the window
4. close MMC and other windows you may have open

Now you have your AD Schema extended. Let's added a SSH public key! Later on you'll need to reconfigure SSHD to look for a key in your Active Directory Schema.

Add SSH Public Key to Active Directory User properties

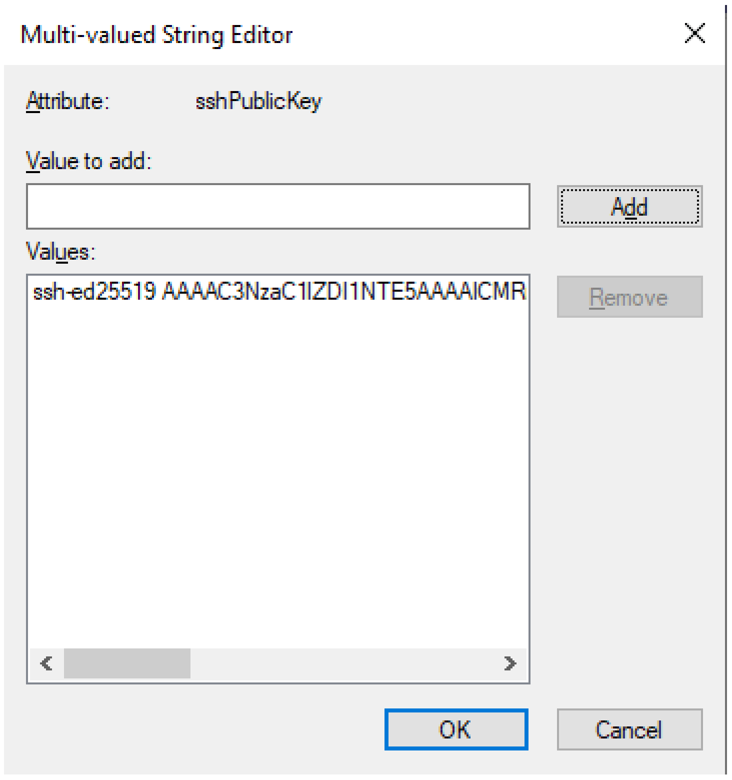
To manually add an SSH public key to a user's properties, start Active Directory Users and Computers. Make sure Advanced Features are enabled under View > Advanced Features.



Enable Advanced Features in Active Directory Users and Computers

Browse to the user for whom you want to add a public SSH key, and follow these steps:

1. Right click the user
2. select Properties
3. select the tab Attribute Editor
4. scroll down to the `sshPublicKey`
5. paste the public key into Value to add field and click Add
6. repeat for a second or third public key if necessary



Attribute: sshPublicKey

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICMR

Add ssh key value to sshPublicKey attribute (on an User object)

7. Close the windows.

Configure OpenSSH AuthorizedKeysCommand

You must have at least OpenSSH 8.6.0.0p1-Beta

Open up the sshd_config_default file. It's location may differ, for me it's in the directory where I installed OpenSSH in Windows Server. Add the following to that file:

```
AuthorizedKeysCommand C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -
NoProfile -NonInteractive -File "c:\path\to\openssh\get-publickey.ps1" -username
%u

AuthorizedKeysCommandUser "system"
```

Here, AuthorizedKeysCommand is the script sshd executes to retrieve the public key. More on this file later. The AuthorizedKeysCommandUser has to be set to System. This is fixed in 8.6.0, hence why you need at least this version.

Save the file and restart `sshd`:

```
Get-Service sshd | Restart-Service
```

AuthorizedKeysCommand PowerShell script to get SSH public key from AD

You need a script that runs an AuthorizedKeysCommand to retrieve the public key from the user's Active Directory sshPublicKeys property. For this, I use PowerShell, you may use something completely different.

```
[CmdletBinding()]
Param(
  [Parameter(Mandatory = $true, Position = 0)]
  [string]$username)$username = $username.Split("\")[1]
  $(
    ([adsisearcher]"(&(objectClass=user)(sAMAccountName=${username}))").FindAll()
| select -Property *
  ).Properties.sshpublickey
```

I chose to use **adsisearcher** because not all servers have the ActiveDirectory PowerShell module available that is required for **Get-ADUser**. In my scenario, the username that tries to authenticate is "domain\user", so I split the username to lookup the username part in Active Directory. The key that is found is retrieved and printed.

Test logging in using SSH keys

OK, you have:

- extended your Active Directory Schema to add support for SSH public keys.
- reconfigured `sshd_config`
- restarted sshd, and
- created a tiny script to pull the public key from the User properties (sshPublicKey attribute)

Now it’s time to test logging in. Ssh to your server and it should automatically log you in.

```
PS C:\Users\janreilink> ssh host.example.org
Microsoft Windows [Version 10.0.17763.3650](c) 2018 Microsoft Corporation. All
rights reserved.

domain\janreilink@host E:\Users\janreilink>
```

This setup is particularly handy if you need to access a Windows Server over ssh from outside its AD domain, but still having a domain user.