

1. How-Tos .....	2
1.1 How-To: Add AD Computer Objects to AD Groups with AD Powershell .....	3
1.2 How-to: Add and Use Custom Search Providers in Chromium Based Web Browsers .....	4
1.3 How-To: AD Powershell   Find all members of an AD group including ForeignSecurityPrincipals .....	10
1.4 How-To: Apply a group policy linked to an AD Security Group without rebooting the server .....	12
1.5 How-To: Capture Network Traffic from a Windows Server without installing Wireshark .....	15
1.6 How-To: Connect to AzureAD with no prompt for creds .....	17
1.7 How-to: Copy Text From Images .....	18
1.8 How-To: Find if a datashare is on-prem or cloud hosted .....	22
1.9 How-To: Find out an Azure Active Directory (AAD) users primary login domain .....	25
1.10 How To: PIMUP via PowerShell .....	27
1.11 How-to: Resolve Date/Time parsing issues in Tomcat on Linux .....	30
1.12 How-To: Search in Powershell history .....	32

# How-Tos

- [How-To: Add AD Computer Objects to AD Groups with AD Powershell](#)
- [How-to: Add and Use Custom Search Providers in Chromium Based Web Browsers](#)
- [How-To: AD Powershell | Find all members of an AD group including ForeignSecurityPrincipals](#)
- [How-To: Apply a group policy linked to an AD Security Group without rebooting the server](#)
- [How-To: Capture Network Traffic from a Windows Server without installing Wireshark](#)
- [How-To: Connect to AzureAD with no prompt for creds](#)
- [How-to: Copy Text From Images](#)
- [How-To: Find if a datashare is on-prem or cloud hosted](#)
- [How-To: Find out an Azure Active Directory \(AAD\) users primary login domain](#)
- [How To: PIMUP via PowerShell](#)
- [How-to: Resolve Date/Time parsing issues in Tomcat on Linux](#)
- [How-To: Search in Powershell history](#)

# How-To: Add AD Computer Objects to AD Groups with AD Powershell

If you have a list of computers that you want to add to a group in AD (e.g. to apply a group policy to a number of machines via a SRV group),

Take a list of computers in a text file .e.g.

## ServerList.Txt

```
cp001wpapp016  
cp001wpapp015  
cp001wpapp019  
cp001wpapp018  
cp001wpapp009  
cp001wpapp010
```

Run the following powershell code:

## Powershell Snippet

```
Get-Content .\ServerList.txt | foreach { Add-ADGroupMember -Server corp.local -Identity SRV_COMP_SRV_CORE_CYB_IFDLAppSupport -Members "$_" }
```

Ensure that the domain is correct for the group and computer objects you are operating on, and ensure to add a \$ to the end of each machine name as computer accounts end in a \$.

If you want to add computers from one machine group to another, use the following code:

```
Add-ADGroupMember -server corptest.local -Identity "SRV_COMP_SRV_CORE_CYB_IFDLObjectivity" -Members (Get-ADGroupMember -server corptest.local -Identity "SRV_COMP_SRV_CORE_CYB_RandC" | Select name | ForEach { $_.Name + "$" }) -Verbose
```

# How-to: Add and Use Custom Search Providers in Chromium Based Web Browsers

- [Background](#)
- [How do I set it up?](#)
- [How do I use it?](#)
- [What providers are available?](#)

## Background

Web browsers that are based on the [Chromium browser engine](#) such as [Google Chrome](#) and the more recent [Microsoft Edge](#) browser have the capability to add custom search providers.

You might assume that these just allow you to choose which search engine to use as your default, however you can also define custom search within different web UIs and custom keywords to direct your search term to a specific provider.

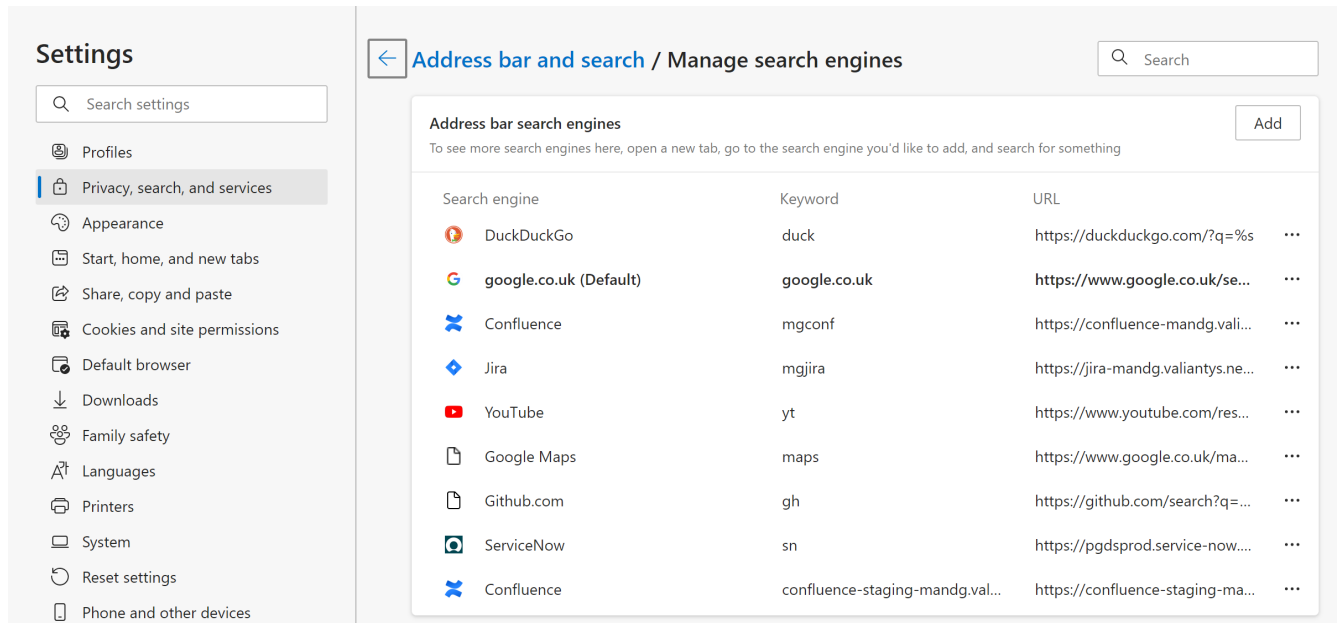
## How do I set it up?

In your browser, goto to the following address in your address bar at the top of the browser: <about://settings/searchEngines>

This will automatically redirect to either <chrome://settings/searchEngines> if you are using Google Chrome, or <edge://settings/searchEngines> if you are using Microsoft Edge.

The rest of the tutorial will use Microsoft Edge as this is the default browser on corporate devices, however the process is largely the same in Google Chrome.

When you go to the search engines settings page, you may see a number of search engines pre-configured e.g.:



In the screenshot above, the ServiceNow, Confluence, Jira and DuckDuckGo providers were added over and above the defaults.

To add one, click on **Add**

You will then be prompted for 3 pieces of information:

×

Add search engine

Search engine

Keyword

URL with %s in place of query

Add

Cancel

Field Name	Description	Example
<b>Search engine</b>	A Name for the search provider	ServiceNow
<b>Keyword</b>	Prefix that, when entered before the search term in the address bar, will direct the search to that provider	sn
<b>URL with %s in place of query</b>	The query URL provided by the web application, with the token %s used to denote where the search term should be added into the URL by the browser.	<a href="https://pgdsprod.service-now.com/nav_to.do?uri=%2F\$sn_global_search_results.do%3Fsysparm_search%3D%s">https://pgdsprod.service-now.com/nav_to.do?uri=%2F\$sn_global_search_results.do%3Fsysparm_search%3D%s</a>

When the details have been entered, click on **Add** again:

Search engine

ServiceNow

Keyword

sn

URL with %s in place of query

https://pgdsprod.service-now.com/nav\_to.do?uri=%2

This will now show in the list of search providers

## How do I use it?

In the search/address bar (sometimes called the OmniBox), enter the keyword and then what you want to search for.

For example, if you want to search ServiceNow for 4Front, you might enter *sn 4Front*:

When you've entered a recognised keyword for a search provider, the right hand side of the OmniBox will tell you to press **Tab** to search that provider:

sn

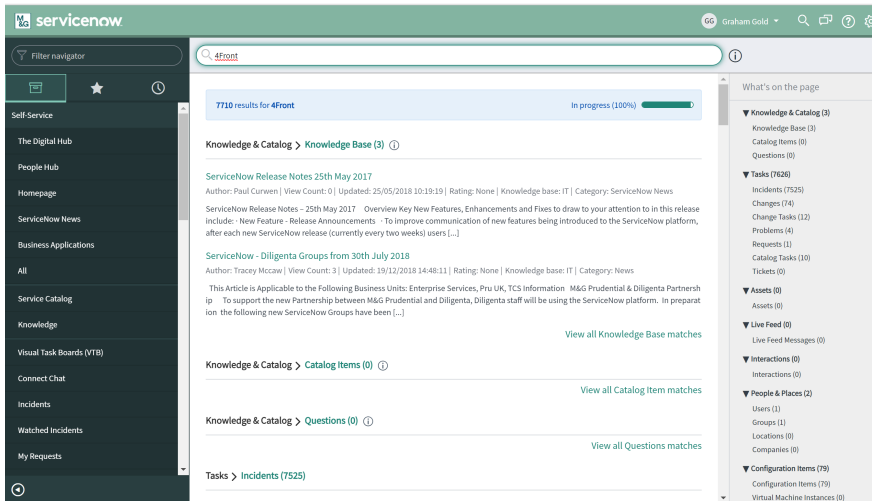
Press Tab to search ServiceNow

You can press tab or just press space and continue entering search text e.g.:

Search ServiceNow | |


Search ServiceNow | 4Front

When you press **Enter** or **Return** you will then get search results direct in the web application e.g.:



## What providers are available?

There are countless providers available. Some that are useful are listed below - please add more as you discover them.

 The keyword can be anything you want - but consider that you might still want to search externally for these e.g. for Confluence and Jira, prefixing the keyword with **mg** still allows easy searching externally for help with those applications.

Search Engine	Keyword	URL with %s in place of query
DuckDuckGo	duck	<a href="https://duckduckgo.com/?q=%s">https://duckduckgo.com/?q=%s</a>
Confluence	mgconf	<a href="https://confluence-mandg.valiantys.net/dosearchsite.action?queryString=%s">https://confluence-mandg.valiantys.net/dosearchsite.action?queryString=%s</a>
Jira	mgjira	<a href="https://jira-mandg.valiantys.net/secure/QuickSearch.jspx?searchString=%s">https://jira-mandg.valiantys.net/secure/QuickSearch.jspx?searchString=%s</a>
YouTube	yt	<a href="https://www.youtube.com/results?search_query=%s&amp;page={startPage?}&amp;utm_source=opensearch">https://www.youtube.com/results?search_query=%s&amp;page={startPage?}&amp;utm_source=opensearch</a>
Google Maps	maps	<a href="https://www.google.co.uk/maps/search/%s?hl=en&amp;source=opensearch">https://www.google.co.uk/maps/search/%s?hl=en&amp;source=opensearch</a>
Github.com	gh	<a href="https://github.com/search?q=%s&amp;ref=opensearch">https://github.com/search?q=%s&amp;ref=opensearch</a>
Stackoverflow.com	stack	<a href="https://stackoverflow.com/search?q=%s">https://stackoverflow.com/search?q=%s</a>
ServiceNow	sn	<a href="https://pgdsprod.service-now.com/nav_to.do?uri=%2F\$sn_global_search_results.do%3Fsysparm_search%3D%s">https://pgdsprod.service-now.com/nav_to.do?uri=%2F\$sn_global_search_results.do%3Fsysparm_search%3D%s</a>
Office365	s365	<a href="https://www.office.com/search?q=%s">https://www.office.com/search?q=%s</a>





# How-To: AD Powershell | Find all members of an AD group including ForeignSecurityPrincipals

If you want to find out members of an AD group using the AD Powershell modules, it is pretty straightforward using the Get-ADGroupMember cmdlet.

However, if the group you query has groups or users nested in from another forest or domain, you will get an error instead of a list of group members:

```
Get-ADGroupMember : An operations error occurred
At line:1 char:1
+ Get-ADGroupMember -server corptest.local -Identity SQLDL_IFDL_Support ...
+ ~~~~~
+ CategoryInfo          : NotSpecified: (SQLDL_IFDL_Support_DBReader:ADGroup) [Get-ADGroupMember], ADException
+ FullyQualifiedErrorId : ActiveDirectoryServer:8224,Microsoft.ActiveDirectory.Management.Commands.GetADGroupMember
```

## Why is this?

Well, as it turns out, AD stores these as **ForeignSecurityPrincipals** - sounds obvious right? It's a Security Principal that is foreign to this domain or forest.

However, AD only stores the **SID** (Security Identifier) of the object, and a few other details but doesn't return the attributes that these cmdlets expect.

And it stores this in the container **ForeignSecurityPrincipals** at the domain root of the domain the FSP is in, not the domain the actual Security Principal resides in e.g. *CN=ForeignSecurityPrincipals,DC=corptest,DC=local*

By default, **Get-ADUser**, **Get-ADGroup** only work on the domain specified by the *-server* parameter - it will not look up information in other domains, so they only return the DN for the FSP object e.g. *CN=S-1-5-21-2093405405-3173940693-2470778517-19427,CN=ForeignSecurityPrincipals,DC=corptest,DC=local*

Worse than that, **Get-ADGroupMember** will fail with the operations error above, because it tries to enumerate attributes that aren't available on the *FSP* object.

## How do I get around this issue?

With a little extra powershell, you can have it strip out the *SID* from the *FSP DN*, and look up that SID in any domain that the domain your logged in user is trusted by.

The code below will do the following:

1. Get all groups in corptest,local domain whose name begins with the string "SQLDL\_IFDL\_Support"
2. For each group
  - a. Write out the group name
  - b. Perform a **Get-ADGroup** for that group and ensure the Members property is returned in the output
  - c. Select and Expand the **Members** property (Which is a collection of objects)
  - d. For Each object in the **Members** collection
    - i. Extract the *SID* from the *DN*
    - ii. Lookup the *SID* using **Get-ADObject** with the *-Filter* parameter - in the filter use **objectSID** instead of *SID*.

```

Get-ADGroup -server corptest.local -filter "Name -like 'SQLDL_IFDL_Support*'" | ForEach {
    Write-Host $_.Name -ForegroundColor Yellow;
    Get-ADGroup -server corptest.local -Identity $_.Name -Properties Members | Select-Object -ExpandProperty Members | ForEach {
        $SID=$_.Split(",")[0].Split("=")[1];
        Get-ADObject -Filter {objectSID -eq $SID}
    }
}

```

This returns output like:

SQLDL_IFDL_Support_Reporting DistinguishedName -----	Name ----	ObjectClass -----
CN=ACC_PAM_Role_IFDLAppSupport,OU=Administrative Groups,OU=Groups,DC=pgds,DC=local	ACC_PAM_Role_IFDLAppSupport	group
CN=ACC_PAM_Role_RandC,OU=Administrative Groups,OU=Groups,DC=pgds,DC=local	ACC_PAM_Role_RandC	group

If you don't need to iterate through a number of groups, the code for a single group is as below (replace server with the FQDN of the domain the group is in, replace Identity with the name of the group):

```

Get-ADGroup -server corptest.local -Identity SQLDL_IFDL_Support_Reporting -Properties Members | Select-Object -ExpandProperty Members | ForEach {
    $SID=$_.Split(",")[0].Split("=")[1];
    Get-ADObject -Filter {objectSID -eq $SID}
}

```

# How-To: Apply a group policy linked to an AD Security Group without rebooting the server

## Introduction

In this article, I'm going to talk about a problem you may have come across before and been frustrated by many times, and how to overcome it.

Something that's very powerful with [Group Policy Objects \(GPO\)](#) is the ability to use [Security Filtering/Delegations](#) to selectively apply (or indeed **not** apply) a GPO to a computer if the computer account is a member of an AD Security Group.

This can allow you to selectively apply an exception to security standards that are enforced via GPO to computers where you need a particular setting **not** to apply.

Another use case is where you are rolling out new security settings in a new GPO and you need the ability to selectively exclude computers from the settings in that policy if a problem is encountered that was caused by that GPO.

## Ok so what's the problem here?

If you make a computer a member of an AD group and then either wait for group policy to refresh (usually every 90-120 minutes) or by forcing a group policy update via the `gpupdate /force` command from an administrative shell, it will often not apply the GPO until the server is rebooted.

This is because the computer knows which groups it is a member of by way of its Kerberos ticket that it gets from AD Domain Controllers at logon and periodically when the ticket expires and is renewed.

Remember that the computer account that you add to the AD group is still just an AD user, just with different attributes than a normal user account. So when the computer is started up it will login to the domain it is joined to just as you do when you login to your domain joined desktop or laptop.

A reboot would renew the ticket from the domain controller and would include an updated group membership list and so that's usually the approach people take.

However, for critical production servers this means downtime for the reboot and therefore an interruption to services.

This may either not be possible until a quiet period or may require change control first, which may delay the application of the GPO (or exclusion, depending upon how you have setup the security filtering for the group in the GPO).

So how can we get around this and get the server to recognise that its group membership has now changed, without having to reboot the server or do anything disruptive?

## So how do I work around the reboot?

You can avoid the need to reboot by renewing the [Kerberos](#) ticket for the computer with the [klist utility](#).

If you were to run `klist sessions | findstr /i %COMPUTERNAME%` at an administrative cmd prompt or `klist sessions | Select-String $env:COMPUTERNAME` from an administrative powershell session, you would see that there are two sessions returned for the computer account.

```
0x3e7 is the logon session.  
0x3e4 is for the network service.
```

These are **always** the same on every Windows computer or server.

So, after you add the computer account to the group that is referenced in the GPO, enter the following from an administrative shell (Powershell, cmd prompt, Windows Terminal): `klist.exe -li 0x3e7 purge`

This will return a response like below:

```
Current LogonId is 0:0xe43f6
Targeted LogonId is 0:0x3e7
    Deleting all tickets:
    Ticket(s) purged!
```

Now, if you run `gpupdate /force` from an administrative shell, it will recognise the computer group membership change and apply the GPOs you were expecting.

Example PowerShell session (truncated) below shows this being done on a server and the resultant output of both **gpupdate** and also **gpresult** which lists the [Resultant Set of Policy \(RSOP\)](#) on the machine:

**After adding server to group CONTOSO\GPO\_Exception\_UserAccountControl and running `gpupdate /force`**

```
gpresult /z
```

```
Microsoft (R) Windows (R) Operating System Group Policy Result tool v2.0
© 2016 Microsoft Corporation. All rights reserved.
```

```
Created on 1/18/2023 at 2:34:23 PM
```

```
RSOP data for CONTOSO\GAIA on TERRA : Logging Mode
```

```
-----
OS Configuration:      Member Server
OS Version:            10.0.14393
Site Name:             HQ
Roaming Profile:       N/A
Local Profile:         C:\Users\GAIA
Connected over a slow link?: No
```

#### COMPUTER SETTINGS

```
-----
CN=TERRA,OU=Servers,OU=Machines,DC=CONTOSO,DC=COM
Last time Group Policy was applied: 1/18/2023 at 2:33:56 PM
Group Policy was applied from:      DC01.CONTOSO.COM
Group Policy slow link threshold:   500 kbps
Domain Name:                       CONTOSO
Domain Type:                       Windows 2008 or later
```

#### Applied Group Policy Objects

```
-----
POL_SRV_BASE
Default Domain Policy
```

#### Purge Kerberos ticket for computer account

```
klist.exe -li 0x3e7 purge
```

```
Current LogonId is 0:0xe43f6
Targeted LogonId is 0:0x3e7
    Deleting all tickets:
    Ticket(s) purged!
```

**Run `gpupdate /force` again**

```
gpupdate /force
```

```
Updating policy...
```

```
Computer Policy update has completed successfully.  
User Policy update has completed successfully.
```

### Check RSoP again

```
gpresult /z
```

```
Microsoft (R) Windows (R) Operating System Group Policy Result tool v2.0  
© 2016 Microsoft Corporation. All rights reserved.
```

```
Created on 1/18/2023 at 2:34:23 PM
```

```
RSOP data for CONTOSO\GAIA on TERRA : Logging Mode
```

```
-----  
OS Configuration:      Member Server  
OS Version:            10.0.14393  
Site Name:             HQ  
Roaming Profile:       N/A  
Local Profile:         C:\Users\GAIA  
Connected over a slow link?: No
```

### COMPUTER SETTINGS

```
-----  
CN=TERRA,OU=Servers,OU=Machines,DC=CONTOSO,DC=COM  
Last time Group Policy was applied: 1/18/2023 at 2:33:56 PM  
Group Policy was applied from:      DCO1.CONTOSO.COM  
Group Policy slow link threshold:   500 kbps  
Domain Name:                        CONTOSO  
Domain Type:                        Windows 2008 or later
```

### Applied Group Policy Objects

```
-----  
POL_Exception_UserAccountControl  
POL_SRV_BASE  
Default Domain Policy
```

And there you have it - we added a computer to a security group that applies a GPO, and didn't have to reboot the computer to make it take effect.

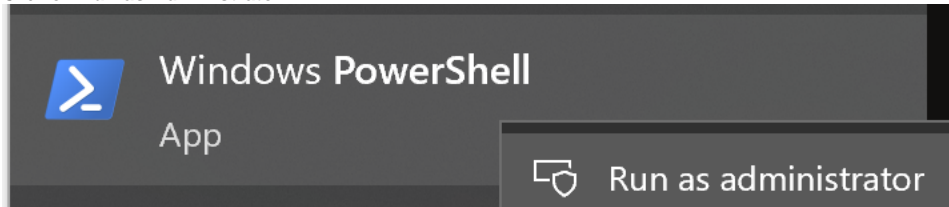
## Summary

In summary, if you have a GPO that is tied to an AD security group and you need it to be applied but don't want to (or perhaps can't) reboot the computer at that time, you can achieve this by purging the Kerberos ticket for the computer account without any downtime or disruption.

# How-To: Capture Network Traffic from a Windows Server without installing Wireshark

If you need to capture network traffic from a Windows server, there is no requirement to install Wireshark on the server - you can use built-in tools to achieve this.

1. Login to server as a user with Administrator permissions
2. Click **Start**, type *powershell* and right click on **Windows PowerShell**
3. Click on **Run as Administrator**



4. Click **Yes** on **User Account Control** prompt
5. Press **Ctrl + Alt** on your keyboard and **Del** on the on-screen keyboard
6. Click **Yes**
7. From the powershell window enter the command below, noting the information in the table below:  
**netsh trace start capture=yes IPv4.Address=<Remote IP> tracefile=<Path>\<filename>.etl**

Parameter	Purpose	Values
IPv4.Address	The IP address of the server communicating with this server whose traffic you wish to capture.	Valid IPV4 IP Address e.g.
	This reduces the amount of packets captured which reduces space usage and aids in easier troubleshooting by filtering out traffic that is not of interest	10.0.0.1
tracefile	Path to and name of the file to write the captured traffic to.	Full file path e.g.
	The file <b>MUST</b> end in the .etl file extension	C:\Users\MyUser\capture.etl

This will overwrite the .etl file if it already exists, will use a maximum capture file size of 250MB and will default to a circular capture e.g. when the file size limit is reached, the oldest packets are removed to make space for new packets.

A full list of syntax and options is available at [Netsh Commands for Network Trace | Microsoft Docs](#)

8. The following response will be returned if trace was successfully started:

Trace configuration:

```
-----  
Status:           Running  
Trace File:       C:\Users\MyUser\capture.etl  
Append:          Off  
Circular:         On  
Max Size:         250 MB  
Report:          Off
```

9. To stop the trace, enter the following syntax : **netsh trace stop**  
You will then see the following messages for several minutes (how long will depend upon a number of factors including service specification, performance, trace file size etc):

```
Correlating traces ... done
Merging traces ... done
Warning: An instance of the 'NT Kernel Logger' is already running.
        System information will not be added to the trace file.
Generating data collection ... done
```

10. When completed the following message will be displayed:

```
The trace file and additional troubleshooting information have been compiled as "C:\Users\MyUser\capture.cab".
File location = C:\Users\MyUser\capture.etl
Tracing session was successfully stopped.
```

11. The default file format is not supported by Wireshark, however you can convert the .etl file to .pcapng format using the [etl2pcapng utility from microsoft that converts an .etl file containing a Windows network packet capture into .pcapng format](#).
12. The latest release can be downloaded as a compiled application from [Releases · microsoft/etl2pcapng · GitHub](#)
13. Once the release is downloaded and extracted from the zip file, the syntax is straightforward:

**etl2pcapng.exe <infile><outfile>**e.g. **etl2pcapng.exe capture.etl out.pcapng**

14. To view the pcapng file on your corporate device, you don't need administrative privileges to install Wireshark, simply request the role **Corp - Wireshark Network Capture Analysis** from [ARC](#) and you will have access to Wireshark, without the network capture driver installed.



## How-To: Connect to AzureAD with no prompt for creds

```
# Connect to Azure AD with no logon prompt

Connect-AzureAD -AccountId 'gary.bond@mandg.com' | Out-Null

# Connect to Azure AD looking up UPN of current user (requires AD module)

$UPN = Get-ADUser -server $env:USERDNSDOMAIN -identity $env:USERNAME | select -ExpandProperty userprincipalname
Connect-AzureAD -AccountId $UPN | Out-Null

# Check for existing AAD session and connect if none

function ConnectAzureAD () {
    $currentSession = $null
    $ErrorActionPreference = 'SilentlyContinue'
    $currentSession = Get-AzureADCurrentSessionInfo
    $ErrorActionPreference = 'Continue'

    if (-not $currentSession) {
        $UPN = Get-ADUser -server $env:USERDNSDOMAIN -identity $env:USERNAME | select -ExpandProperty userprincipalname
        Connect-AzureAD -AccountId $UPN | Out-Null
    }
}
```

# How-to: Copy Text From Images

- [Background](#)
- [How do I do this?](#)
- [How do I use it?](#)
- [Tips on Usage](#)

## Background

Sometimes you need to provide evidence or write documentation and you don't have access to the end system, so all you have are screenshots.

However, having the text from the screenshot would improve the quality of the documentation - especially extracting configuration names, commands etc.

## How do I do this?

By default, you have access to part of the Microsoft Office suite called Microsoft OneNote.

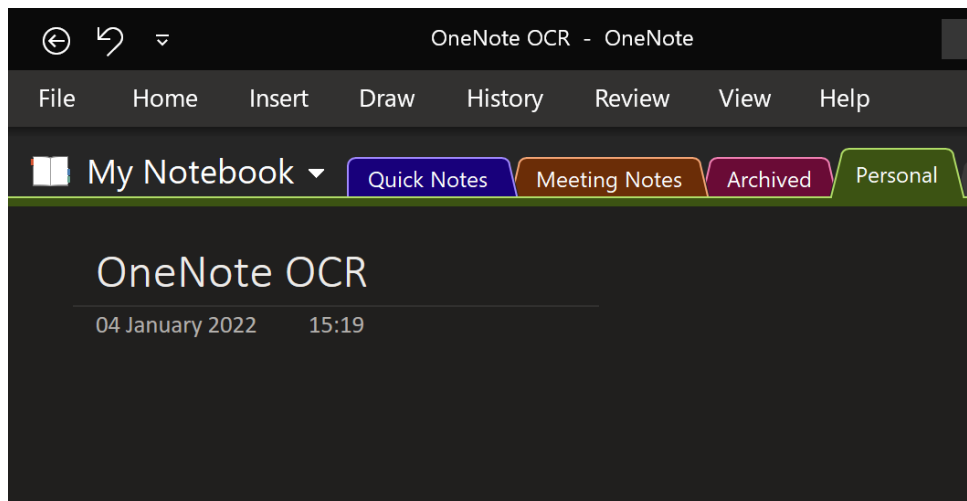
OneNote is a very useful note taking tool and is a much better experience than having lots of disparate documents saved in various places (word documents, text files etc).

It has some very handy features including a global search, multiple workbooks, sections within workbooks etc for organisation.

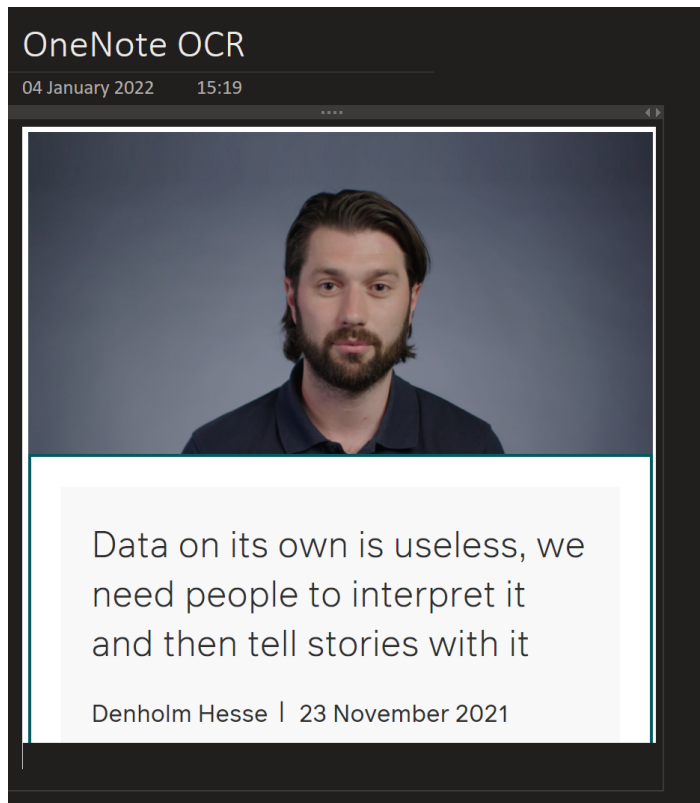
However, one of its lesser known features is that it has OCR (Optical Character Recognition) built-in and it is a completely offline feature - no internet connection required and no privacy concerns around data leakage.

## How do I use it?

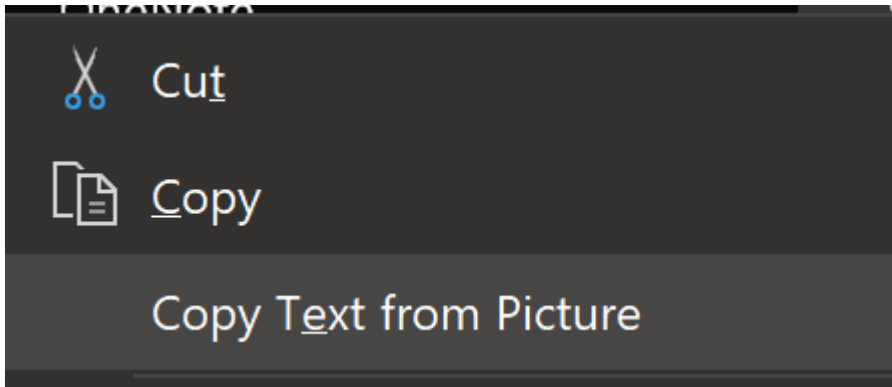
In a new or existing note page:



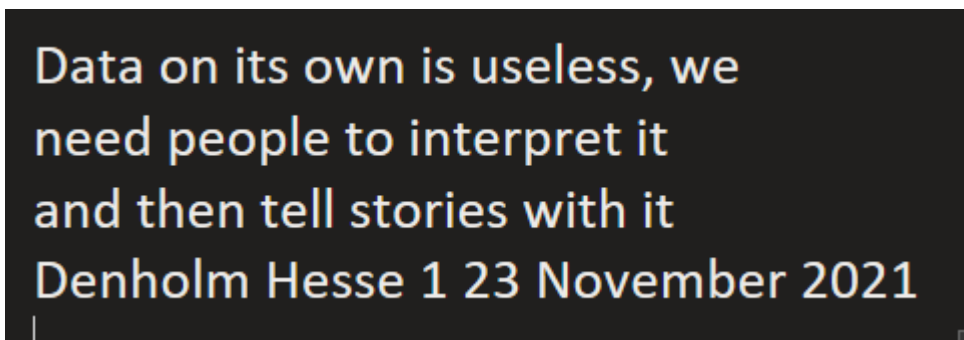
Paste an image into the page:



Right click on the image and select **Copy Text from Picture**:



You then have text you can paste into another document e.g.:



## Tips on Usage

The quality of the extracted text varies mainly by the image size/resolution/quality.

Generally speaking the better the image quality, the larger the text in the image, the more successful the text extraction will be.

The formatting may not always be perfect, but it will at least give you the text so you can edit/use as needed.

**Copyrighted Material**

As ever - be mindful that you must not use copyrighted material or otherwise infringe any laws in the use of this functionality.

For that reason, the example chosen above is from the <https://www.mandgplc.com/> public facing website.

If it is permitted to distribute the content with appropriate attribution to the content owner/creator, always ensure that this is honoured and visible in any content you produce using it.

# How-To: Find if a datashare is on-prem or cloud hosted

## Problem

With increased use of DFS for datashares, it is not always clear where a datashare is actually hosted.

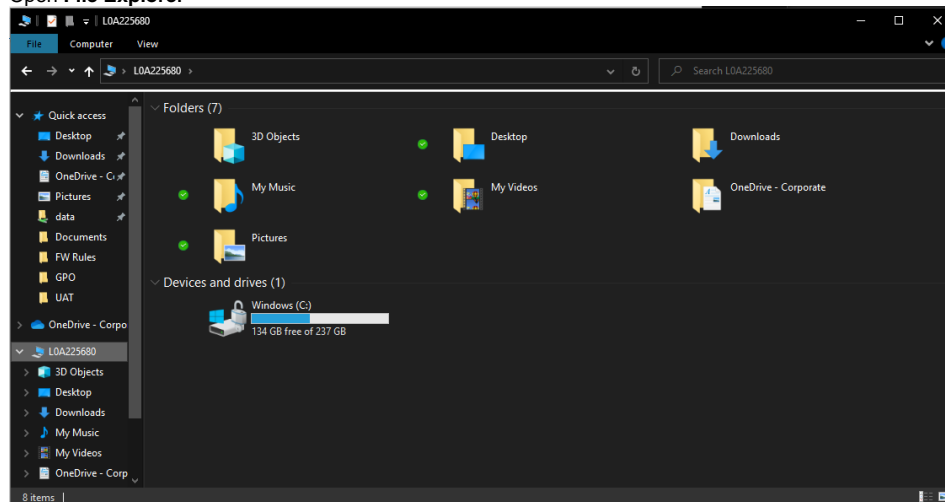
For example, if it is using DFS namespaces you are likely accessing it via the domain Fully Qualified Domain Name (FQDN) e.g. `\\corp.local\DATA\path\to\share`.

## Solution.

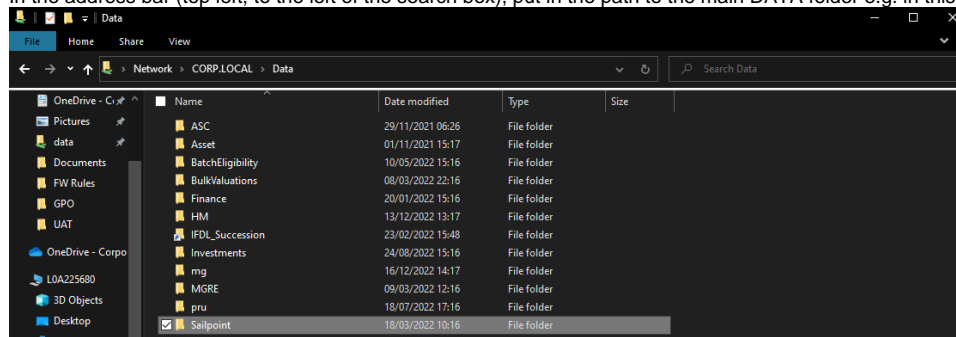
Most top-level data folders are permissioned to be visible to all, but the child folders (the shared folders) are where the permissions are (or should be) applied and therefore won't be visible unless you have access.

So, for example, if you wanted to know where the share `\\corp.local\DATA\Sailpoint\Dev` is hosted:

### 1. Open File Explorer

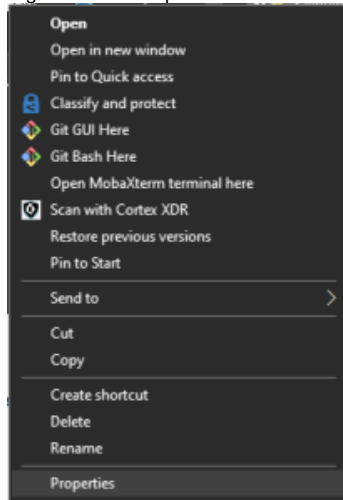


### 2. In the address bar (top left, to the left of the search box), put in the path to the main DATA folder e.g. in this case, `\\corp.local\Data`:

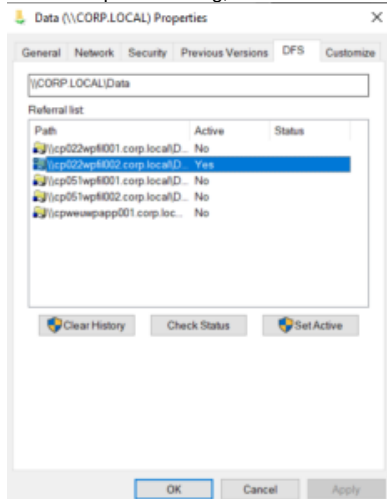


3. Press **Enter**

4. Right click on the parent folder for the data shares, in this case, **Sailpoint** and click on **Properties**

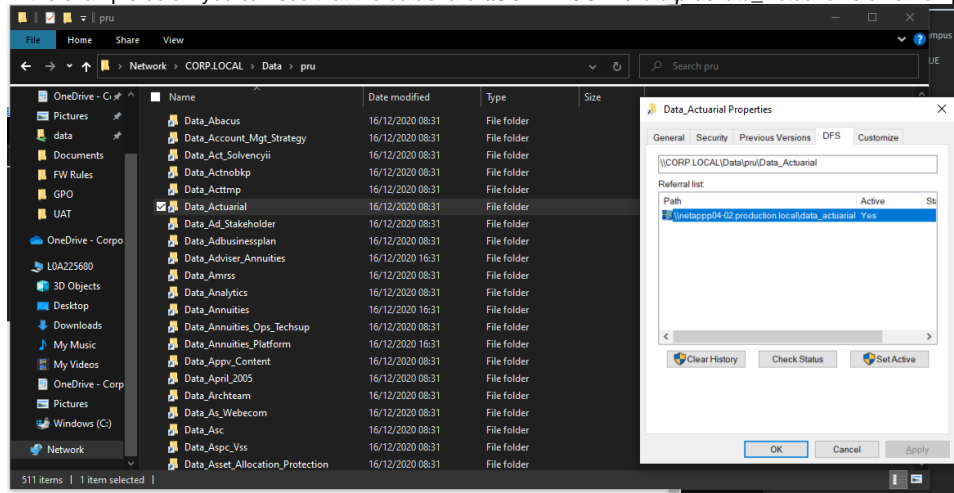


5. On the Properties dialog, click on the **DFS** tab

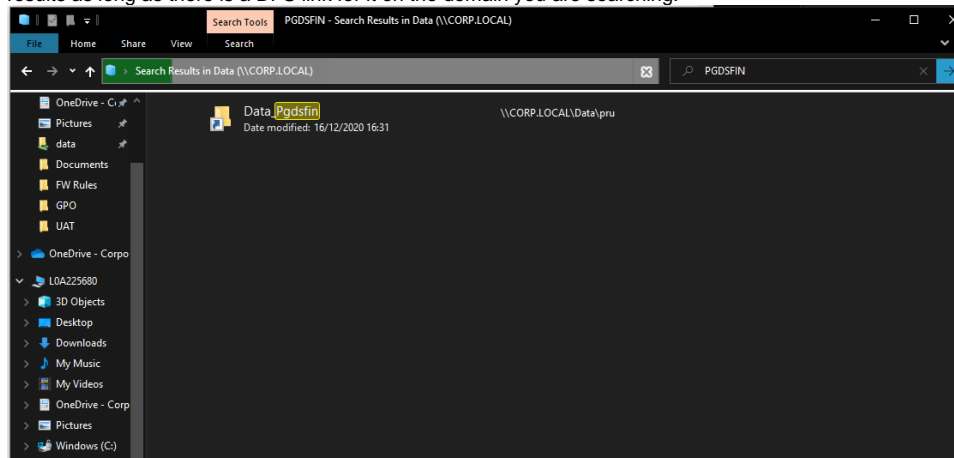


6. In this case you can see that there are several DFS links, to both on-premise and Azure hosted file servers (cp<site number> prefix is on-premise, CPWEU prefix is Azure, WestEUrope region (See [PRJ0026985 - Server Naming Standards](#) for more information on server naming standards)

7. In the example below you can see that the datashare `\\CORP.LOCAL\Data\pru\Data_Actuarial` is on an on-premise netapp fileserver



8. If you don't know the path and only the datashare name, you can also just navigate to `\\<DOMAIN FQDN>\Data` and then search for the share name (which is really a shared folder) and you will find it in the search results as long as there is a DFS link for it on the domain you are searching:





# How-To: Find out an Azure Active Directory (AAD) users primary login domain

If you have an Azure Active Directory user, synced in from an on-premise Active Directory Domain Services (ADDS) domain, it can be tricky to know which domain they were synced into AAD from - e.g. which is their primary user login domain.

As long as you have read access to AAD, you can run this powershell code to determine the primary login domain if the user's primary domain was CORP.LOCAL, PRODUCTION.LOCAL, MANDG.CO.UK or PGDS.LOCAL:

```
Param
(
    [Parameter(Mandatory = $true)]
    [String] $searchUser
)

$mgSID      = (Get-ADDomain -server mandg.co.uk).DomainSID.Value
$prodSID    = (Get-ADDomain -server production.local).DomainSID.Value
$corpSID    = (Get-ADDomain -server corp.local).DomainSID.Value
$pgdsSID    = (Get-ADDomain -server pgds.local).DomainSID.Value
$corpTenant = "aa42167d-6f8d-45ce-b655-d245ef97da66"

try {
    $ADTenant = Get-AzureADTenantDetail
}
catch [Microsoft.Open.Azure.AD.CommonLibrary.AadNeedAuthenticationException] {
    Write-Host "You're not connected to AzureAD, authenticating...";
    Connect-AzureAD -TenantID $corpTenant -Verbose;
}

$userSID = (Get-AzureADUser -SearchString $searchUser).OnPremisesSecurityIdentifier
If ($userSID.Length -gt 0){
    switch -wildcard ($userSID) {
        "$corpSID*"      { return "[INFO] User $searchuser Primary Domain is CORP.LOCAL" }
        "$prodSID*"     { return "[INFO] User $searchuser Primary Domain is PRODUCTION.LOCAL" }
        "$mgSID*"       { return "[INFO] User $searchuser Primary Domain is MANDG.CO.UK" }
        "$pgdsSID*"     { return "[INFO] User $searchuser Primary Domain is PGDS.LOCAL" }
        Default         { return "[ERROR] User SID $userSID not found in the domains CORP.LOCAL, PRODUCTION.LOCAL, MANDG.CO.UK or PGDS.LOCAL" }
    }
} else {
    return "[ERROR] User $searchUser not found in AzureAD in Tenant: $($ADTenant.DisplayName)"
}
```

This code is maintained here: [Get-AzureADUserPrimaryDomain.ps1 - Repos](#)

It is run as follows:

```
\Get-AzureADUserPrimaryDomain.ps1 -searchUser "Gold, Graham"
```

And returns output such as:

User Gold, Graham Primary Domain is CORP.LOCAL

It performs the following steps:

1. Connect to AzureAD
2. Query the user supplied (in format ***Surname, FirstName***) in Azure AD and store the **OnPremisesSecurityIdentifier** attribute value
3. Compare that against the known Security Identifier (SID) prefixes for CORP, PRODUCTION, PGDS and MG domains.
4. If it matches one, return that domain as the Primary Domain.

# How To: PIMUP via PowerShell

The following will connect to Azure PIM (Corporate tenant), check what roles you are entitled to elevate to, and allow elevation. This is instant and doesn't have the same waiting period like the Portal.

The duration of elevation is set to 1 hour.

This uses the AD module to look up the UPN of the the current logged on user but can be adapted if you are elevating another account e.g. PGDS.

Very rarely MFA will be prompted and you will have to use the Portal.

```
PS C:\> pimup

Connecting to Azure...

Current eligible roles:
1. Authentication Administrator
2. User Administrator
3. Application Administrator
4. Conditional Access Administrator
5. Groups Administrator
6. Privileged Role Administrator
Enter role selection : 3

Enter reason for privileged access: confluence sso cert
If you get an ["MfaRule"] error here, you will need to use the Azure Portal to PIM Up :
https://portal.azure.com/#blade/Microsoft\_Azure\_PIMCommon/ActivationMenuBlade/aadmigratedroles/provider/aadroles

ResourceId      : aa42167d-6f8d-45ce-b655-d245ef97da66
RoleDefinitionId : 9b895d92-2cd3-44c7-9d02-a6ac2d5ea5c3
SubjectId       : 89b43fb8-4236-4e09-981d-ac0e34f7646c
Type            : UserAdd
AssignmentState  : Active
Schedule        : class AzureADMSPrivilegedSchedule {
                  StartDateTime: 22/12/2021 16:07:37
                  EndDateTime: 22/12/2021 17:07:35
                  Type: Once
                  Duration: PT0S
                  }

Reason          : confluence sso cert
```

```
function ConnectAzureAD () {
    $currentSession = $null
    $ErrorActionPreference = 'SilentlyContinue'
    $currentSession = Get-AzureADCurrentSessionInfo
    $ErrorActionPreference = 'Continue'
```

```

    if (-not $currentSession) {
        $script:UPN = Get-ADUser -server $env:USERDNSDOMAIN -identity $env:USERNAME | select -ExpandProperty userprincipalname
        Connect-AzureAD -AccountId $UPN | Out-Null
    }
}

function PIMUP () {
    Write-Host "Connecting to Azure..."

    # Connect to Azure and get UPN of current user
    ConnectAzureAD
    if (-not $UPN) {
        $UPN = Get-ADUser -server $env:USERDNSDOMAIN -identity $env:USERNAME | select -ExpandProperty userprincipalname
    }
    $tenantID = "aa42167d-6f8d-45ce-b655-d245ef97da66"
    $userID = (Get-AzureADUser -ObjectId $UPN).ObjectID

    # Get eligible roles and display as menu of options
    $roles = Get-AzureADMSPrivilegedRoleAssignment -ProviderId "aadRoles" -ResourceId $tenantID -Filter "subjectId eq '$userID'" | where {$_.AssignmentState -eq "Eligible"} |
    foreach-object { Get-AzureADMSPrivilegedRoleDefinition -ProviderId "aadRoles" -ResourceId $tenantID -Id $_.RoleDefinitionId }

    Write-Host "`nCurrent eligible roles:"
    $menu = @{}
    for ($i=1;$i -le $roles.count;$i++)
    { Write-Host "$i. $($roles[$i-1].DisplayName)"
      $menu.Add($i,($roles[$i-1].Id))}

    [int]$choice = Read-Host 'Enter role selection '
    if ($choice -notmatch '^[1-] + [regex]::escape($roles.count) + ']+$') {
        Write-host 'Selection is not valid...'
        PIMUP
    }
    else {
        $selection = $menu.Item($choice)
        Write-Host ""
        [string]$Reason = Read-Host -Prompt "Enter reason for privileged access"
        Write-Host 'If you get an ["MfaRule"] error here, you will need to use the Azure Portal to PIM Up :' -ForegroundColor Yellow
        Write-Host "https://portal.azure.com/#blade/Microsoft_Azure_PIMCommon/ActivationMenuBlade/aadmigratedroles/provider/aadroles" -ForegroundColor Yellow
        #Calculates end time, 1 hour from now
        $schedule = New-Object Microsoft.Open.MSGraph.Model.AzureADMSPrivilegedSchedule
        $schedule.Type = "Once"
        $schedule.StartDateTime = (Get-Date).ToUniversalTime().ToString("yyyy-MM-ddTHH:mm:ss.fffZ")
        $schedule.EndDateTime = (Get-Date $schedule.StartDateTime).AddHours(1)

        Open-AzureADMSPrivilegedRoleAssignmentRequest -ProviderId 'aadRoles' -ResourceId $tenantID -RoleDefinitionId $selection -SubjectId $userID -Type 'UserAdd'
        -AssignmentState 'Active' -schedule $schedule -reason $Reason | out-host
    }
}

```



# How-to: Resolve Date/Time parsing issues in Tomcat on Linux



**Apache Tomcat** is an open source web server for Java programming that is developed and maintained by the Apache software foundation.

It runs on both Windows and Linux Operating Systems.

When migrating a Tomcat application from Windows to Linux, it is worth noting how Tomcat handles timezones, as this may cause you issues with data handling/parsing in your applications.

Tomcat uses the Operating System timezone settings.

In Windows, this means the timezone set in the Date and Time settings e.g.

## Date & time

### Current date and time

14:03, 08 September 2021

#### Synchronize your clock

Last successful time synchronization: 08/09/2021 14:02:46

Time server: PGTVDOMP01.pgds.local

Sync now

#### Time zone

(UTC+00:00) Dublin, Edinburgh, Lisbon, London

Adjust for daylight saving time automatically



On

This means that it will automatically handle clock changes (daylight savings time).

However, in Linux, it defaults to UTC as that is the default for Linux and is independent of any setting applied via the **tzselect** or **timedatectl set-timezone** commands.

Therefore you need to tell Tomcat to behave accordingly by amending the **catalina.sh** config script that is part of Tomcat and is usually located in `/usr/local/tomcat<version>/bin/` e.g. `/usr/local/tomcat9/bin/`

Add the 2 lines below after the first line of the file:

```
TOMCAT_TIMEZONE="-Duser.timezone=Europe/London"
CATALINA_OPTS="$CATALINA_OPTS $TOMCAT_TIMEZONE"
```

This can be done via nano, vi or a linux text editor of your choice.

The file, when saved, should start as below:

```
#!/bin/sh
TOMCAT_TIMEZONE="-Duser.timezone=Europe/London"
CATALINA_OPTS="$CATALINA_OPTS $TOMCAT_TIMEZONE"

# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

For the changes to take effect - restart Tomcat on the server via the following commands:

```
sh /usr/local/tomcat9/bin/shutdown.sh

sh /usr/local/tomcat9/bin/startup.sh
```

# How-To: Search in Powershell history

It's common to use command history in Powershell to find and execute commands previously issued - either to speed up your workflow or to avoid typing errors when repeating long commands.

However, whilst it's common to just use the **up** and **down** arrow keys to work backwards and forwards through the history, there are more efficient ways to find something in history:

## History CmdLets

PowerShell has the following cmdlets that allow you to view your history and repeat previously executed commands:

**Get-History** returns all history based on profile settings for how many commands to keep in history e.g.:

```
Get-History

Id CommandLine
--
1 nslookup wl-lds-systest.pre-release.local
2 test-netconnection fnweuwpads051.cloud.pgds.local -port 3389
3 Get-ItemProperty -Path "Registry::HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings"
```

You can then run a particular command again with **Invoke-History** e.g. if you want to run command ID 3 again to check internet settings in the registry:

```
Invoke-History -Id 3

#Returns the following:

Get-ItemProperty -Path "Registry::HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings"

DisableCachingOfSSLPages : 0
IE5_UA_Backup_Flag       : 5.0
PrivacyAdvanced           : 1
SecureProtocols           : 2688
User Agent                : Mozilla/4.0 (compatible; MSIE 8.0; Win32)
CertificateRevocation     : 1
PrivDiscUiShown          : 1
EnableHttp1_1             : 1
ProxyHttp1.1              : 1
AutoConfigURL             : http://pgdspac.pgds.local/proxymg_nonWSS.pac
ProxyOverride             : *.local;<local>
EnableNegotiate           : 1
MigrateProxy              : 1
ProxyEnable               : 0
ZonesSecurityUpgrade      : {25, 39, 36, 132...}
WarnonZoneCrossing        : 0
```



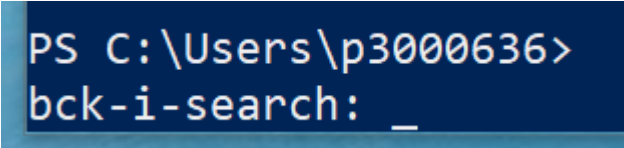
```
LockDatabase      : 132622724891281161
AutoDetect        : 0
PSPath            : Microsoft.PowerShell.Core\Registry::HKCU\Software\Microsoft\Windows\CurrentVersion\Internet
                  Settings
PSParentPath       : Microsoft.PowerShell.Core\Registry::HKCU\Software\Microsoft\Windows\CurrentVersion
PSChildName        : Internet Settings
PSProvider         : Microsoft.PowerShell.Core\Registry
```

### Reverse Search

If you have a large history, that could still take a while to find what you are looking for (although you could pass the output to other cmdlets to search the output). You also still have to run at least 2 commands, one to view history and one to invoke history.

However, if you press the keyboard combination of **ctrl + r** this invokes the reverse feature.

You will then see the prompt **bck-i-search:**



```
PS C:\Users\p3000636>
bck-i-search: _
```

As you type in search text it will suggest the best match e.g.:



```
PS C:\Users\p3000636> Get-ItemProperty -Path "Registry::HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings"
bck-i-search: internet_
```

Pressing **Enter** or **Return** will then execute the command e.g.

```

PS C:\Users\p3000636> Get-ItemProperty -Path "Registry::HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings"
DisableCachingOfSSLPages : 0
IE5_UA_Backup_Flag       : 5.0
PrivacyAdvanced          : 1
SecureProtocols          : 2688
User_Agent               : Mozilla/4.0 (compatible; MSIE 8.0; Win32)
CertificateRevocation    : 1
PrivDiscUiShown         : 1
EnableHttp1.1            : 1
ProxyHttp1.1             : 1
AutoConfigURL            : http://pgdspac.pgds.local/proxymg_mg_nonWSS.pac
ProxyOverride            : *.local;<local>
EnableNegotiate          : 1
MigrateProxy             : 1
ProxyEnable              : 0
ZonesSecurityUpgrade     : {25, 39, 36, 132...}
WarnonZoneCrossing       : 0
LockDatabase             : 132622724891281161
AutoDetect               : 0
PSPath                   : Microsoft.PowerShell.Core\Registry::HKCU\Software\Microsoft\Windows\CurrentVersion\Internet
                           Settings
PSParentPath              : Microsoft.PowerShell.Core\Registry::HKCU\Software\Microsoft\Windows\CurrentVersion
PSChildName               : Internet Settings
PSProvider                : Microsoft.PowerShell.Core\Registry

```

Pressing **Tab** instead will just add the command text to the command prompt, allowing you to edit it before executing by pressing **Enter** or **Return** when ready:

```

PS C:\Users\p3000636> Get-ItemProperty -Path "Registry::HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings

```