



Home > > Microsoft Defender Advanced Threat  
Protection  
> > Microsoft Defender  
ATP

[Back to Blog](#)[< Newer Article](#)[Older Article >](#)

Raviv Tamir Microsoft

01-28-2019 12



## WDATP API "Hello World" (or using a simple PowerShell script to pull alerts via WDATP APIs)



5 Minutes



Low complexity

Applying a security solution in an enterprise environment can be a complex endeavor. Customers deploy various layers of protection solutions, investigation platforms and hunting tools. Security Operation teams attempt to tackle this task, but typically lack expensive and experienced human resources to overcome this challenge. Automation is a decent mitigation but automating the security procedures and wiring the security components all together to a solid cyber security solution, requires programmatic access to each solution.

In these series of blogs, we will walk you through common automation scenarios that you can achieve with Windows Defender ATP to optimize workflows. For more information on Windows Defender ATP APIs, see [the full documentation](#).



We called this blog "Hello World" as every long software journey starts with a simple step. We'll show you how to programmatically extract Windows Defender ATP alerts with a PowerShell script. You can schedule this script to run on any machine and you may modify it to use the alert information in your specific use case. We can imagine a handful of standard use cases where a Security Operations Center (SOC) can leverage this basic capability.

Some scenarios where this can be applied include use with security information and event management (SIEM) connectors, ticketing systems, and security orchestration and response (SOAR) solutions. SIEM connectors may be the simplest example while ticketing systems are a common one, and SOAR solutions may be a complex use case.

So let gets started....

How long it takes to go through this example?

It only takes 5 minutes done in two steps:

- Application registration: takes 2 minutes
- Use examples: only requires copy/paste of a short PowerShell script



**Do I need permission to connect?**

For the app registration stage, you must have a Global administrator role in your Azure Active Directory (Azure AD) tenant.

**Step 1 - Register the app in Azure Active Directory**




- With your Global administrator credentials, login to the Azure portal.
  - **Azure Active Directory > App registrations > New registration.**

The screenshot shows the Azure portal interface for 'WcdTestPrd - App registrations'. The left sidebar contains navigation links, with 'Azure Active Directory' highlighted. The main content area features a search bar and a table of applications. The table has two columns: 'DISPLAY NAME' and 'APPLICATION (CLIENT) ID'. The search results show 'MDATP is the best!' under the 'DISPLAY NAME' column, and 'No results' is displayed below the table.

- In the registration form:
  - **Name** - Name your application.
  - **Supported account type** – leave the default setting.
  - **Redirect Uri** – leave empty.




[Home](#) > [WcdTestCloudCus - App registrations](#) >  Microsoft [Registration](#)

## Register an application

**\* Name**

The user-facing display name for this application (this can be changed later).

### Supported account types

Who can use this application or access this API?

☒ Accounts in this organizational directory only 

☐ Accounts in any organizational directory

☐ Accounts in any organizational directory and personal Microsoft accounts (e.g. Skype, Xbox, Outlook.com)

[Help me choose...](#)

### Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web 

*e.g. https://myapp.com/auth*

[By proceeding, you agree to the Microsoft Platform Policies](#) 

[Register](#)



- Click **Register**

The application I created is the authentication entity, just like a service account. I now need to set permissions to my app and save its credential for later use.

- On your new application page, click **APIs** > **Add permission** > **APIs my organization uses** > type **WindowsDefenderATP** and click on **WindowsDefenderATP**

**Note:** WindowsDefenderATP does not appear in the original list. You need to start writing its name in the text box to see it appear

Home > YourAppName - API permissions

YourAppName - API permissions

Search (Ctrl+/)

Overview

Quickstart

Manage

Branding

Authentication

Certificates & secrets

**API permissions**

Expose an API

Owners

Manifest

Support + Troubleshooting

Troubleshooting

New support request

API permissions

Applications are authorized to use APIs by requesting grant/deny access.

**Add a permission**

API / PERMISSIONS NAME

Microsoft Graph (1)

User.Read

These are the permissions that this application requires. See the documentation for more information.

Grant consent

As an administrator, you can grant consent on behalf of users. Granting consent on behalf of users means that end users will not be shown a consent dialog.

**Grant admin consent for WcdTestPrd**

Request API permissions

Select an API

Microsoft APIs **APIs my organization uses** My APIs

Apps in your directory that expose APIs are shown below


WindowsDefenderATP

NAME	APPLICATION (CLIENT) ID
WindowsDefenderATP	fc780465-2017-40d4-a0c5-307022471b92
WindowsDefenderATPCustomerTiConnector	ec3d9e43-f260-4375-ad5a-160032eeff25
WindowsDefenderATPSiemConnector	5f19cce1-79ca-4a23-aa2b-b487f78ef0a0


- Choose **Application permissions** > **Alert.Read.All** > Click on **Add permissions**:



# Request API permissions



[All APIs](#)

**WindowsDefenderATP**  
<https://api-uk.securitycenter.microsoft.com>

What type of permissions does your application require?

Delegated permissions  
Your application needs to access the API as the signed-in user.


Application permissions  
Your application runs as a background service or daemon without a signed-in user.

Select permissions [expand all](#)

PERMISSION	ADMIN CONSENT REQUIRED
▶ AdvancedQuery	
▼ Alert (1)	
<input checked="" type="checkbox"/> Alert.Read.All Read all alerts ⓘ	Yes
<input type="checkbox"/> Alert.ReadWrite.All Read and write all alerts ⓘ	Yes
▶ Event	
▶ File	
▶ Ip	
▶ Machine	
▶ SecurityRecommendation	
▶ Ti	

Add permissions

Discard



- After clicking the **Add Permissions** button, on the next screen we need to grant consent for the permission to take effect.

Home > WcdTestCloudCus - App registrations > YourAppName - API permissions

Microsoft

Search (Ctrl+J)

Permissions have changed. Users and/or admins will have to consent even if they have already done so previously.

### API permissions

Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users are given the opportunity to grant/deny access.

[+ Add a permission](#)

API / PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED
▼ Microsoft Graph (1)			
User.Read	Delegated	Sign in and read user profile	-
▼ WindowsDefenderATP (1)			
Alert.Read.All	Application	Read all alerts	Yes ⚠ Not granted for WcdT...

These are the permissions that this application requests statically. You may also request user consent-able permissions dynamically through code. [See best practices for requesting permissions](#)

### Grant consent

As an administrator, you can grant consent on behalf of all users in this directory. Granting admin consent for all users means that end users will not be shown a consent screen when using the application.

[Grant admin consent for WcdTestCloudCus](#)

- Press the "Grant admin consent for {your tenant name}" button.

As explained, the registered app is an authentication entity with permission to access all alerts for reading. Now I need to get and store the authentication and authorization credentials:

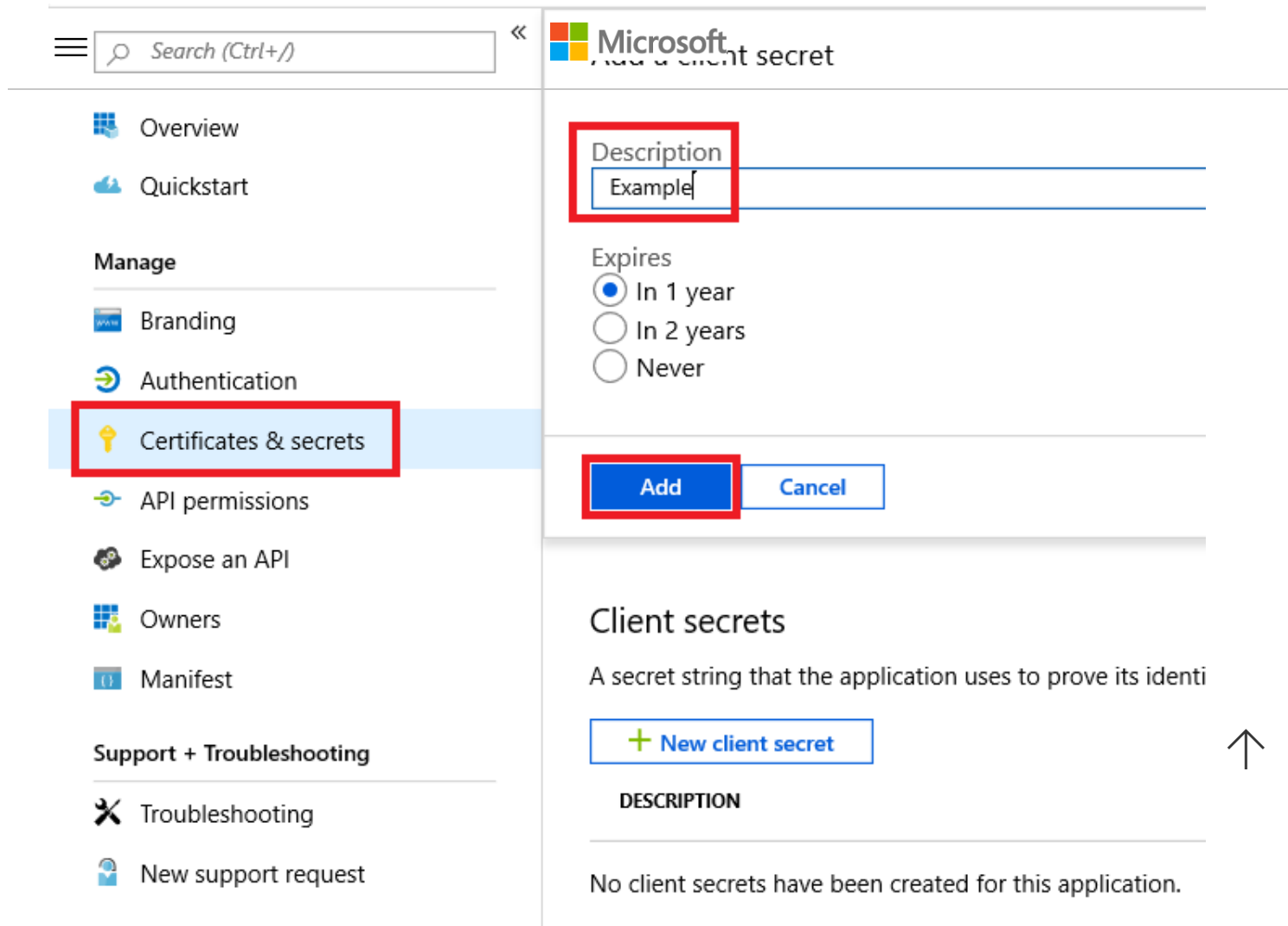


- Key (application secret), Application ID, and Tenant ID

Think of your secret like a **password**, Application ID as **username** and Tenant ID as a **domain**.

- To get credentials:
  - In your application page, Click **Certificate & Secrets**
  - Specify a key description and set an expiration for 1 year.





The screenshot shows the Microsoft Defender ATP console. On the left, the 'Certificates & secrets' menu item is highlighted with a red box. In the main area, the 'Add a client secret' form is visible. The 'Description' field contains the text 'Example' and is also highlighted with a red box. Below the description field, there are radio buttons for 'Expires' with options 'In 1 year' (selected), 'In 2 years', and 'Never'. At the bottom of the form, the 'Add' button is highlighted with a red box. Below the form, the 'Client secrets' section is shown, which includes a '+ New client secret' button and a table with one row for a new client secret. The table has a header 'DESCRIPTION' and a body row with the text 'No client secrets have been created for this application.'

- Click **Add**.
- The application key will appear.  
IMPORTANT: **Copy and store this key in a safe place. Treat it like a password.**
- Get your **application ID** and your **tenant ID**:
  - On your application page, go to **Overview** and copy the following







Home > WcdTestPrd - App registrations > YourAppName

## YourAppName

Delete Endpoints

Overview

Quickstart

### Manage

Branding

Authentication

Certificates & secrets

API permissions



Welcome to the new and improved App registrations. Looking to learn how it's

Display name : YourAppName

Application (client) ID : 94fbf312-ae3f-47f4-8eeb-d375d1299e8f

Directory (tenant) ID : f839b112-d9d7-4d27-9bf6-94542403f21c

Object ID : b687654e-064c-4514-b48c-3fd5aa009a1f

**Done!** You have successfully registered an application. You may reuse this application when going through the exercises that we'll be using in future blogs and experiments.



Now we'll need to connect the API which means getting a token. The token is proof for Windows Defender ATP that an API call is authenticated and authorized.

### Connecting the API:

- Copy the text below to PowerShell ISE or to a text editor.



## That code get the App Context Token and  name "Latest-token.txt" under the current di  
 # Paste below your Tenant ID, App ID and App Secret (App key).

```
$tenantId = 'f839b112-d9d7-94d27-9bf6-94542403f21c' ### Paste your own tenant ID here
$appId = '94fbf312-ae3f-47f4-8eeb-d375d1299e8f' ### Paste your own app ID here
$appSecret = 'ABCDEFGHJKLMNOPQRSTUVWXYZ1234567890=' ### Paste your own app keys here

$resourceAppIdUri = 'https://api.securitycenter.windows.com'
$oAuthUri = "https://login.windows.net/$TenantId/oauth2/token"

$authBody = [Ordered] @{
    resource = "$resourceAppIdUri"
    client_id = "$appId"
    client_secret = "$appSecret"
    grant_type = 'client_credentials'
}

$authResponse = Invoke-RestMethod -Method Post -Uri $oAuthUri -Body $authBody -ErrorAction Stop
$token = $authResponse.access_token
Out-File -FilePath "./Latest-token.txt" -InputObject $token
return $token
```



- Save the script to file. You can name it "**Get-Token.ps1**".
- Running this script by pressing F5 will get a token and save it in the working folder under the name "**./Latest-token.txt**".



Note:

If you are getting an error:

**\Get-Token.ps1 cannot be loaded because running scripts is disabled on this system. For more information, see about\_Execution\_Policies at <https://go.microsoft.com/fwlink/?LinkID=135170>.**

That error indicates that your Powershell execution policy not allowing you to run scripts. You can change the execution policy by running that command in Powershell console:

```
PS c:\> Set-ExecutionPolicy unrestricted -Scope CurrentUser
```

Consider consulting with your system administrator about your organization's Powershell execution policy.

## Sanity Check



- In your browser go to: <https://jwt.ms/>
  - Copy the token (the content of the Latest-token.txt file).
  - Paste in the top box.
  - Look for the "roles" section. Find the Alert.Read.All role.
- 
- Now let's get the alerts, Copy the following text to a new PowerShell Script.

```
# Returns Alerts created in the past 48 hours.
$token = ./Get-Token.ps1
#run the script Get-Token.ps1 - make sure you are running this script from the same folder of Get-Tok
# Get Alert from the last 48 hours. Make sure you have alerts in that time frame.

$dateTime = (Get-Date).ToUniversalTime().AddHours(-48).ToString("o")

# The URL contains the type of query and the time filter we create above
# Read more about other query options and filters here
$url = "https://api.securitycenter.windows.com/api/alerts?`$filter=alertCreationTime ge $dateTime"

# Set the WebRequest headers
$headers = @{
    'Content-Type' = 'application/json'
    Accept = 'application/json'
    Authorization = "Bearer $token"
}

# Send the webrequest and get the results.
$response = Invoke-WebRequest -Method Get -Uri $url -Headers $headers -ErrorAction Stop
#Extract the alerts from the results.
$alerts = ($response | ConvertFrom-Json).value | ConvertTo-Json

#Get string with the execution time. We concatenate that string to the output file to avoid overwrite
$dateTimeForFileName = Get-Date -Format o | foreach {$_ -replace ":", "."}

#save the result as json and as csv
$outputJsonPath = "./Latest Alerts $dateTimeForFileName.json"
$outputCsvPath = "./Latest Alerts $dateTimeForFileName.csv"

Out-File -FilePath $outputJsonPath -InputObject $alerts
($alerts | ConvertFrom-Json) | Export-CSV $outputCsvPath -NoTypeInfoation
```

- Save the file in the same folder you save the script (Get-Token.ps1).
- You can run the script by right-clicking on the file and choosing "Run with PowerShell" or run it from PowerShell console.

**NOTE: If you are using powershell\_ise.exe make sure to change directory to the directory with the scripts.**

- You will now see two files (json and csv) created in the same folder as the scripts.
- The files are the latest alert from your tenant in the past 48 hours.

You're all done! You have just successfully:

- Created and registered an application
- Granted permission for that application to read alerts
- Connected the API
- Use a PowerShell script to return alerts created in the past 48 hours

**In the next blog, we'll walk you through updating alert status programmatically. I invite you to suggest more use cases that you'd like for us to blog about, provide feedback, and ask questions about this post!**

Thanks!

[@Haim Goldshtein](#), security software engineer, WDATP

[@Dan Michelson](#), program manager, WDATP

[@Ben Alfasi](#), software engineer, Windows Defender ATP



Tags: Alert API hello PowerShell wdatp world

9 Likes

Share

17 Comments



PhotM Phantom of the Mobile New Contributor

01-31-2019 01:1



This maybe naive since I don't remember seeing anything, as I scrolled through, about the "Defender" Module. It is a great Module and I believe it can be used remotely or at the very least on a providing server in a small operation?

I use it all the time from Powershell ISE.

0 Likes



**Dan Michelson** Microsoft

02-04-2019 03:1

Thanks for your feedback.

Happy to hear that you are using the "Defender" module. The "Defender" module covers the AV management. Here, in this blog post, we are taking the first step in encouraging our customers to try and automate WDATP procedures. Will be great to get feedback about that too. We are here for your questions and feedback.

1 Like



**Murray Wall** Contributor

02-19-2019 12:1

I get this back in the Token

```
"roles": [  
  "Alert.Read.All"  
],
```

but this from the Invoke

Invoke-WebRequest : The remote server returned an error: (401)  
Unauthorized.

Did I copy the pwd key wrong, I was sure I didn't... Thoughts Ideas?

0 Likes

**Dan Michelson** Microsoft

02-19-2019 02:0

Hi,

From first look, it seems that the part of granting permissions was not completed successfully. Please jump to the line "In this example, we only need the "Read all alerts" permission." and try to follow the steps there. If you find that the desired read permission is missing, that might be the problem. Please, pay attention to the "grant permissions" step.

Hope this helps.

0 Likes

**Haim Goldshtein** Microsoft

02-20-2019 12:3

Hi Murray,

it looks like your token holds the right permission.

first, please pay attention that you save both the Get-token.ps1 and Get-Alert.ps1 in the same folder.

when you open the script with Powershell\_ise.exe the default folder is the user folder (PS C:\Users\{Username}>) which means that if you run the script by pressing F5 the Powershell executer will try to find the file Get-Token.ps1 under your user profile.

you need to change the running directory to the directory you saved the scripts or to call Get-Token.ps1 script by using its full path and not the relative path.

Hope it helps.

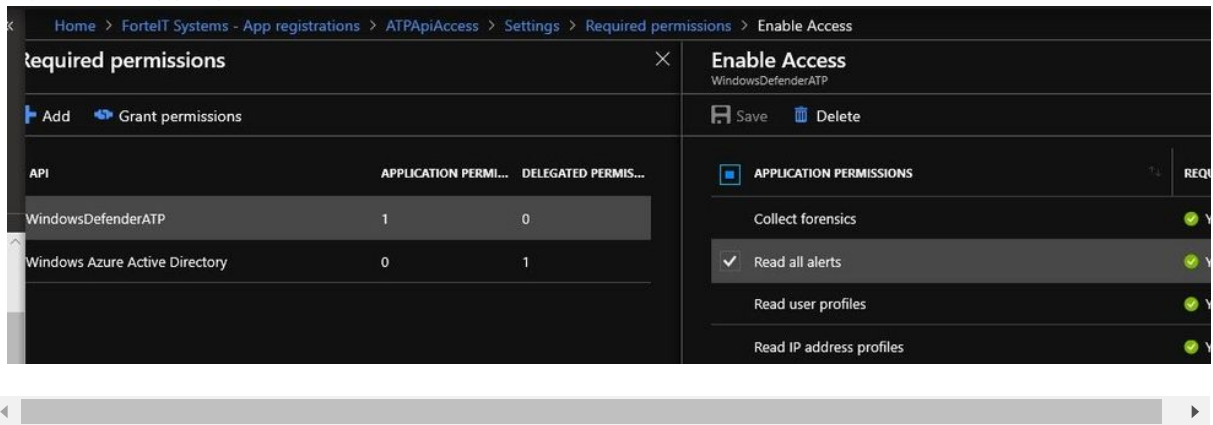
0 Likes

**Murray Wall** Contributor

02-20-2019 05:0



I went back in a validated my step in shot, and confirmed I am in the right directory when running the script, the token looks ok, - What license for Azure AD do I need to get the logging?



0 Likes



Haim Goldshtein Microsoft

02-20-2019 05:33

according to the token validation result you published, you don't have any problem with the permission.

could you make sure that you are running the script while the powershell console running directory point to the folder with the scripts?

for example, if you saved the scripts under C:\temp\api then you need to make sure to change the directory in the powershell running folder (you can use the dir command to check if you see the scripts)

```

21
22 $outputJsonPath = "./Latest Alerts $dateTimeForFileName.json"
23 $outputCsvPath = "./Latest Alerts $dateTimeForFileName.csv"
24
25 Out-File -FilePath $outputJsonPath -InputObject $alerts
26 ($alerts | ConvertFrom-Json) | Export-Csv $outputCsvPath -NoTypeInformation

```

```

PS C:\temp\api> dir

Directory: C:\temp\api

Mode                LastWriteTime         Length Name
----                -
-a----             1/17/2019   2:24 PM           884 Get-Alerts.ps1
-a----             1/24/2019  11:06 AM           886 Get-Token.ps1


```

```

PS C:\temp\api>

```



if you run the script from another  Alert.ps1 script will not find the Get-Token.ps1 script and you will get an error message 401 (unauthorized) like you published in your first message.

```
PS C:\temp> C:\temp\blog\Get-Alerts.ps1
.\get-token.ps1 : The term '.\get-token.ps1' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spe
At C:\temp\blog\Get-Alerts.ps1:14 char:10
+ $token = .\get-token.ps1
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (.\get-token.ps1:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

Invoke-WebRequest : The remote server returned an error: (401) Unauthorized.
At C:\temp\blog\Get-Alerts.ps1:16 char:13
+ $response = Invoke-WebRequest -Method Get -Uri $url -Headers $headers ...
+ ~~~~~
+ CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebException
+ FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

ConvertFrom-Json : Cannot bind argument to parameter 'InputObject' because it is null.
At C:\temp\blog\Get-Alerts.ps1:18 char:25
+ $alerts = ($response | ConvertFrom-Json).value | ConvertTo-Json
+ ~~~~~
+ CategoryInfo          : InvalidData: (:) [ConvertFrom-Json], ParameterBindingValidationException
+ FullyQualifiedErrorId : ParameterArgumentValidationErrorNullNotAllowed,Microsoft.PowerShell.Commands.ConvertFromJsonCommand

PS C:\temp>
```

Hope it helps.



0 Likes



02-20-2019 06:0



**Murray Wall** Contributor

```
PS C:\WINDOWS\system32> cd \scripts
PS C:\scripts> $token = .\Get-Token.ps1
PS C:\scripts> $dateTime = (Get-Date).ToUniversalTime().AddHours(-48).ToString("o")
PS C:\scripts> $url = "https://api.securitycenter.windows.com/api/alerts?"$filter=alertCreationTime ge $dateTime"
PS C:\scripts> $headers = @{
    >> 'Content-Type' = 'application/json'
    >> Accept = 'application/json'
    >> Authorization = "Bearer $token"
    >> }
PS C:\scripts> $response = Invoke-WebRequest -Method Get -Uri $url -Headers $headers -ErrorAction Stop
Invoke-WebRequest : The remote server returned an error: (401) Unauthorized.
At line:1 char:14
+ ... $response = Invoke-WebRequest -Method Get -Uri $url -Headers $headers ...
+ ~~~~~
+ CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebExc
ption
+ FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

PS C:\scripts> .\Get-Token.ps1
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IlzeE1KTUxDSURXTVRQd1pS5jZ0eC1DRHh3MCIisImtpZCI6Ii1zeE1KTUxDSURXTVRQd1pS5jZ0eC1DRHh3MCMjcy_eyJhdWQiOiJodHRwczovL2FwaS5zZWNIcm10eWNIbnRlciIsIiwiaWF0IjE5MTA2NDAzMTEyOTYyNTUwNjc0OjE1ODkpnWUFvY3RlcnmRUUVYdGlXS9qdVhRdTdHQUFBPStImFwcGklIjoibWYxMTJjNGItTzB7MS00Nzh1ThmoTQtM2InNyJfKNTZkZmU4Iiwia
```

Yes, the article made that clear so I even when down to the command line and ran the commands. The token is getting generated, its the web response that is being tagged as unauth - I looked at the token and it looks fine. Its why I was asking about the AAD License required..

Thanks for you help and responses on this matter - I understand this is just a training item and shouldn't be so hard to run - so the overall importance isn't great , just trying to figure out why its not working has me digging!





0 Likes



**Murray Wall** Contributor

02-24-2019 07:0

Great update and realization for me - seems counter intuitive for me to say this, but you obviously (or not so to me!) need to be licensed for ATP via M365 license prior to running the code, If you have a M365 E5 license or have ATP licenced you can run the code, or alternatively you can license ATP for a trial, here <https://www.microsoft.com/en-us/windowsforbusiness/windows-atp>

Without being licensed for ATP you will not be authorized to run the demonstration!

Thanks and keep the posts coming - its time to license up some features to see all the great features

Murray



3 Likes



**mwatter90** Frequent Visitor

02-27-2019 03:5

I'm getting the same unauthorized message. If the application is the one obtaining the token that give you access to the API then how do you apply a license to that application? I have all appropriate licenses assigned to my admin account, and I still get the unauthorized message.

0 Likes



**Murray Wall** Contributor

02-27-2019 06:0

Hello, one of the troubleshooting steps I was asked to do first was to validate my ATP via this page here -



to <https://securitycenter.windows.microsoft.com/defender/atp/integration>. This will validate your ATP license is good and you have the necessary access to run the script. I did not as I do not have an E5 or M365 license with ATP enabled. If you get the following graphic then I would investigate w



## ! No subscriptions found

Before you can start using Windows Defender Advanced Threat Protection, you need to subscribe to the service. See [Windows Defender ATP product site](#) or contact your Microsoft account team for information.

### Already subscribed to a trial or commercial license?

Windows Defender ATP license settlement can take up to 30 minutes. Please try to log in again later.

Need further assistance? [Contact support](#)

[Click here to retry now](#)

ith your licensing team . Let me know how you make out!



1 Like



**mwatter90** Frequent Visitor

02-28-2019 12:00

I'm able to view everything in the portal. I'm still getting unauthorized when I execute the script. I gave my application all possible permissions to the defender api. I'll probably just open a ticket with MS.

0 Likes



**Dan Michelson** Microsoft

03-04-2019 03:10

To assist we need more details.  
We'll contact you.



0 Likes

03-05-2019 09:2

Deleted Not applicable

I was able to execute everything with no issues but when the results come back they are missing the machine name and the user name associated with the alert. The results return the machine id but we are not sure how to get that converted to machine name.

0 Likes

03-05-2019 11:3



Dan Michelson Microsoft

We'll sort this out with you in the private channel and publish insights here if applicable.

0 Likes

03-06-2019 05:0




Haim Goldshtein Microsoft

Hey @Deleted ,

As you probably saw, the alert includes only machine id and if you want to enrich that data with machine name you need to call another API method [Get machine by ID](#) which returns lots of information about the machine.

you can see in our second blog [Ticketing system integration](#) how we pull alerts and iterate through the alerts to use the alert id to update the ticketing system. you can use the same logic to get the machine information and enrich the alert. your script can look like:



# Returns Alerts created in the  Microsoft

# Setting a place holder for a code to open a ticket in external ticketing system.

```
$token = .\Get-Token.ps1
```

```
$dateTime = (Get-Date).ToUniversalTime().AddHours(-400).ToString("o")
```

```
$url = "https://api.securitycenter.windows.com/api/alerts?`$filter=alertCreationTime ge $dateTime"
```

```
$headers = @{
    'Content-Type' = 'application/json'
    Accept = 'application/json'
    Authorization = "Bearer $token"
}
```

```
$response = Invoke-RestMethod -Method Get -Uri $url -Headers $headers -ErrorAction Stop
```

```
#foreach alert, get the machineId and alertId and isolate machine while writing the alert
```

```
foreach ($alert in $response.value)
```

```
{
    $alertId = $alert.id
    $machineId = $alert.machineId
```

```
$url = "https://api.securitycenter.windows.com/api/machines/$machineId"
```

```
$body = @{}
```

```
$headers = @{
    Authorization = "Bearer $token"
}
```

```
$machineInfoResponse = Invoke-WebRequest -Method Get -Uri $url -Body $body -Header $headers
```

```
#check the isolatino request code and write to log file.
```

```
if($machineInfoResponse.StatusCode -eq 200) {
    $machineInfo = $machineInfoResponse | select -Expand Content | ConvertFrom-Json
    $machineName = $machineInfo.computerDnsName
```

```
    # replace the next line with your code to take the alerts data and enrich it
    [System.Windows.MessageBox]::Show("Alert ID - $alertId. MachineName - $machineName")
}
```

```
else {
    [System.Windows.MessageBox]::Show("Failed to get machine info for machineId - $machineId")
}
```





```
}  
}
```



We are working on a Powershell module which should ease the use of the API from Powershell.

Please reply if the answer answers your issue.

0 Likes

**Deleted** Not applicable

03-07-2019 10:1

Perfect.. Thanks @Haim Goldshtein... One last question as I was going through all the ms docs on api's. We came across <https://wdatp-alertexporter-us.windows.com/api/alerts> and when we try to call it we. I'm assuming this link isn't valid anymore?

Authorization has been denied for this request

0 Likes

You must be a registered user to add a comment. If you've already registered, sign in. Otherwise, register and sign in.

[Comment](#)

Microsoft

Popular

Windows Dev Center

Microsoft Azure

Microsoft Visual Studio

Office Dev Center

asp.net

IIS.net

Learning Resources

Channel 9

Windows Development Videos

Microsoft Virtual Academy

Programs

App Developer Agreement

Windows Insider Program

Microsoft Affiliate Program

BizSpark (for startups)

Microsoft Imagine

For IT Pros

Microsoft Power BI

Microsoft SQL Server

Internet of Things

Operations Management Suite

Values

Diversity and inclusion

Accessibility

Environment

Microsoft Philanthropies

Corporate Social Responsibility

Privacy at Microsoft

Company

Careers


About Microsoft


Company news

Investors

Research

Site map

 English (United States)



Contact us

Privacy & cookies

Terms of use

Trademarks

About our ads

© 2019 Microsoft