JEA Session Configurations

07/10/20199 minutes to read  

In this article

A JEA endpoint is registered on a system by creating and registering a PowerShell session configuration file. Session configurations define who can use the JEA endpoint and which roles they have access to. They also define global settings that apply to all users of the JEA session.

Create a session configuration file

To register a JEA endpoint, you must specify how that endpoint is configured. There are many options to consider. The most important options are:

- Who has access to the JEA endpoint
- Which roles they be assigned
- Which identity JEA uses under the covers
- The name of the JEA endpoint

These options are defined in a PowerShell data file with a `.pssc` extension known as a PowerShell session configuration file. The session configuration file can be edited using any text editor.

Run the following command to create a blank template configuration file.

PowerShell

Copy

```
New-PSSessionConfigurationFile -SessionType
RestrictedRemoteServer -Path
.\MyJEAEndpoint.pssc
```

Tip

Only the most common configuration options are included in the template file by default. Use the `-Full` switch to include all applicable settings in the generated PSSC.

The `-SessionType RestrictedRemoteServer` field indicates that the session configuration is used by JEA for secure management. Sessions of this type operate in **NoLanguage** mode and only have access to the following default commands (and aliases):

- Clear-Host (cls, clear)
- Exit-PSSession (exsn, exit)
- Get-Command (gcm)
- Get-FormatData
- Get-Help
- Measure-Object (measure)
- Out-Default
- Select-Object (select)

No PowerShell providers are available, nor are any external programs (executables or scripts).

For more information about language modes, see about_Language_modes.

Choose the JEA identity

Behind the scenes, JEA needs an identity (account) to use when running a connected user's commands. You define which identity JEA uses in the session configuration file.

Local Virtual Account

Local virtual accounts are useful when all roles defined for the JEA endpoint are used to manage the local machine and a local administrator account is sufficient to run the commands successfully. Virtual accounts are temporary accounts that are unique to a specific user and only last for the duration of their PowerShell session. On a member server or workstation, virtual accounts belong to the local computer's **Administrators** group. On an Active Directory Domain Controller, virtual accounts belong to the domain's **Domain Admins** group.

PowerShell Copy

```
# Setting the session to use a virtual ac-
count
RunAsVirtualAccount = $true
```

If the roles defined by the session configuration don't require full administrative privilege, you can specify the security groups to which the virtual account will belong. On a member server or workstation, the specified security groups must be local groups, not groups from a domain.

When one or more security groups are specified, the virtual account isn't assigned to the local or domain administrators group.

PowerShell Copy

```
# Setting the session to use a virtual ac-
count that only belongs to the NetworkOpera-
tor and NetworkAuditor local groups
RunAsVirtualAccount = $true
RunAsVirtualAccountGroups =
'NetworkOperator', 'NetworkAuditor'
```

Note

Virtual accounts are temporarily granted the Logon as a service right in the local server security policy. If one of the VirtualAccountGroups specified has already been granted this right in the policy, the individual virtual account will no longer be added and removed from the policy. This can be useful in scenarios such as domain controllers where revisions to the domain controller security policy are closely audited. This is only available in Windows Server 2016 with the November 2018 or later rollup and Windows Server 2019 with the January 2019 or later rollup.

Group-managed service account

A group-managed service account (GMSA) is the appropriate identity to use when JEA users need to access network resources such as file shares and web services. GMSAs give you a domain identity that is used to authenticate with resources on any machine within the domain. The rights that a GMSA provides are determined by the resources you're accessing. You don't have admin rights on any machines or services unless the machine or service administrator has explicitly granted those rights to the GMSA.

PowerShell

Copy

```
# Configure JEA sessions to use the GMSA in
the local computer's domain
# with the sAMAccountName of 'MyJEAGMSA'
GroupManagedServiceAccount = 'Domain\MyJEAGM-
SA'
```

GMSAs should only be used when necessary:

- It's difficult to trace back actions to a user when using a GMSA. Every user shares the same run-as identity. You must review PowerShell session transcripts and logs to correlate individual users with their actions.
- The GMSA may have access to many network resources that the connecting user doesn't need access to. Always try to limit effective permissions in a JEA session to follow the principle of least privilege.

> **Note**
>
> Group managed service accounts are only available on domain-joined machines using PowerShell 5.1 or newer.

For more information about securing a JEA session, see the security considerations article.

Session transcripts

It's recommended that you configure a JEA endpoint to automatically record transcripts of users' sessions. PowerShell session transcripts contain information about the connecting user, the run as identity assigned to them, and the commands run by the user. They can be useful to an auditing team who needs to understand who made a specific change to a system.

To configure automatic transcription in the session configuration file, provide a path to a folder where the transcripts should be stored.

PowerShell Copy

```
TranscriptDirectory = 'C:\ProgramData\JEACon-
figuration\Transcripts'
```

Transcripts are written to the folder by the **Local System** account, which requires read and write access to the directory. Standard users should have no access to the folder. Limit the number of security administrators that have access to audit the transcripts.

User drive

If your connecting users need to copy files to or from the JEA endpoint, you can enable the user drive in the session configuration file. The user drive is a **PSDrive**that is mapped to a unique folder for each connecting user. This folder allows users to copy files to or from the system without giving them access to the full file system or exposing the FileSystem provider. The user drive contents are persistent across sessions to accommodate situations where network connectivity may be interrupted.

PowerShell Copy

```
MountUserDrive = $true
```

By default, the user drive allows you to store a maximum of 50MB of data per user. You can limit the amount of data a user can consume with the *UserDriveMaximumSize* field.

PowerShell Copy

```
# Enables the user drive with a per-user lim-
it of 500MB (524288000 bytes)
MountUserDrive = $true
UserDriveMaximumSize = 524288000
```

If you don't want data in the user drive to be persistent, you can configure a scheduled task on the system to automatically clean up the folder every night.

> **Note**
>
> The user drive is only available in PowerShell 5.1 or newer.

For more information about PSDrives, see Managing PowerShell drives.

Role definitions

Role definitions in a session configuration file define the mapping of **users** to **roles**. Every user or group included in this field is granted permission to the JEA endpoint when it's registered. Each user or group can be included as a key in the hashtable only once, but can be assigned multiple roles. The name of the role capability should be the name of the role capability file, without the `.psrc` extension.

PowerShell Copy

```
RoleDefinitions = @{
    'CONTOSO\JEA_DNS_ADMINS'     = @{ RoleCa-
pabilities = 'DnsAdmin', 'DnsOperator', 'Dns-
Auditor' }
    'CONTOSO\JEA_DNS_OPERATORS' = @{ RoleCa-
pabilities = 'DnsOperator', 'DnsAuditor' }
```

```
    'CONTOSO\JEA_DNS_AUDITORS'  = @{ RoleCa-
pabilities = 'DnsAuditor' }
}
```

If a user belongs to more than one group in the role definition, they get access to the roles of each. When two roles grant access to the same cmdlets, the most permissive parameter set is granted to the user.

When specifying local users or groups in the role definitions field, be sure to use the computer name, not **localhost** or wildcards. You can check the computer name by inspecting the `$env:COMPUTERNAME` variable.

PowerShell

Copy

```
RoleDefinitions = @{
    'MyComputerName\MyLocalGroup' = @{ Role-
Capabilities = 'DnsAuditor' }
}
```

Role capability search order

As shown in the example above, role capabilities are referenced by the base name of the role capability file. The base name of a file is the filename without the extension. If multiple role capabilities are available on the system with the same name, PowerShell uses its implicit search order to select the effective role capability file. JEA does **not** give access to all role capability files with the same name.

JEA uses the `$env:PSModulePath` environment variable to determine which paths to scan for role capability files. Within each of those paths, JEA looks for valid PowerShell modules that contain a "RoleCapabilities" subfolder. As with importing

modules, JEA prefers role capabilities that are shipped with Windows to custom role capabilities with the same name.

For all other naming conflicts, precedence is determined by the order in which Windows enumerates the files in the directory. The order isn't guaranteed to be alphabetical. The first role capability file found that matches the specified name is used for the connecting user. Since the role capability search order isn't deterministic, it's **strongly recommended**that role capabilities have unique filenames.

Conditional access rules

All users and groups included in the **RoleDefinitions**field are automatically granted access to JEA endpoints. Conditional access rules allow you to refine this access and require users to belong to additional security groups that don't impact the roles to which they're assigned. This is useful when you want to integrate a just-in-time privileged access management solution, smartcard authentication, or other multifactor authentication solution with JEA.

Conditional access rules are defined in the RequiredGroups field in a session configuration file. There, you can provide a hashtable (optionally nested) that uses 'And' and 'Or' keys to construct your rules. Here are some examples of how to use this field:

PowerShell Copy

```
# Example 1: Connecting users must belong to
a security group called "elevated-jea"
RequiredGroups = @{ And = 'elevated-jea' }

# Example 2: Connecting users must have
signed on with 2 factor authentication or a
smart card
# The 2 factor authentication group name is
```

```
"2FA-logon" and the smart card group name is
"smartcard-logon"
RequiredGroups = @{ Or = '2FA-logon', 'smart-
card-logon' }

# Example 3: Connecting users must elevate
into "elevated-jea" with their JIT system and
have logged on with 2FA or a smart card
RequiredGroups = @{ And = 'elevated-jea', @{
Or = '2FA-logon', 'smartcard-logon' }}
```

Note

Conditional access rules are only available in PowerShell 5.1 or newer.

Other properties

Session configuration files can also do everything a role capability file can do, just without the ability to give connecting users access to different commands. If you want to allow all users access to specific cmdlets, functions, or providers, you can do so right in the session configuration file. For a full list of supported properties in the session configuration file, run `Get-Help New-PSSessionConfigurationFile -Full`.

Testing a session configuration file

You can test a session configuration using the Test-PSSessionConfigurationFile cmdlet. It's recommended that you test your session configuration file if you've manually edited the `.pssc` file. Testing ensures the syntax is correct. If a session configuration file fails this test, it can't be registered on the system.

Sample session configuration file

The following example shows how to create and validate a session configuration for JEA. The role definitions are created and stored in the `$roles` variable for convenience and readability. It isn't a requirement to do so.

PowerShell

Copy

```
$roles = @{
    'CONTOSO\JEA_DNS_ADMINS'     = @{ RoleCa-
pabilities = 'DnsAdmin', 'DnsOperator', 'Dns-
Auditor' }
    'CONTOSO\JEA_DNS_OPERATORS' = @{ RoleCa-
pabilities = 'DnsOperator', 'DnsAuditor' }
    'CONTOSO\JEA_DNS_AUDITORS'  = @{ RoleCa-
pabilities = 'DnsAuditor' }
}

New-PSSessionConfigurationFile -SessionType
RestrictedRemoteServer -Path .\JEAConfig.pssc
-RunAsVirtualAccount -TranscriptDirectory
'C:\ProgramData\JEAConfiguration\Transcripts'
-RoleDefinitions $roles -RequiredGroups @{ Or
= '2FA-logon', 'smartcard-logon' }
Test-PSSessionConfigurationFile -Path .\JEA-
Config.pssc # should yield True
```

Updating session configuration files

To change the properties of a JEA session configuration, including the mapping of users to roles, you must unregister.

Then, re-register the JEA session configuration using an updated session configuration file.