HOW TO

Set up a Devolutions Server data source with Remote Desktop Manager PowerShell Module

Devolutions | WHERE IT MEETS SECURITY

Remote Desktop Manager offers an extensive PowerShell module, not only to manage Devolutions Server, but also the Remote Desktop Manager (RDM) client itself. In fact, you can provision an RDM client almost entirely from the PowerShell command-line, without any GUI interactions at all!

The version of Remote Desktop Manager in use is 2022.1.27.0.

Installing the Remote Desktop Manager PowerShell Module

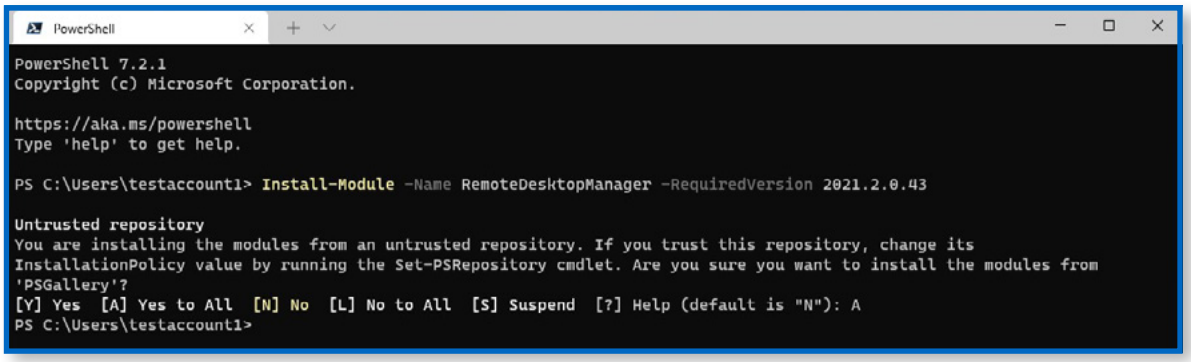
Before you configure RDM, you must first install the Remote Desktop Manager PowerShell module itself from the PowerShell Gallery.

1. Launch PowerShell 7, optionally as an Administrator, and in this article PowerShell 7.2.1 running in Windows Terminal is used.

2. Install the latest version of the RemoteDesktopManager package, which at the time of this writing is **2021.2.0.43.** Choose either, **[Y] Yes** or **[A] Yes to All** (the latter of the two means you will not be prompted for future module installs from this repository), when prompted to install from the PowerShell Gallery.

```
Install-Module -Name RemoteDesktopManager -RequiredVersion 2021.2.0.43
```

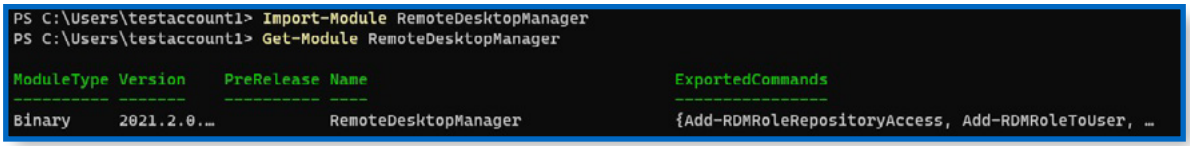
To install the latest version, simply omit the -RequiredVersion parameter. Additionally, you can choose to install the module for the current user only with the -Scope CurrentUser parameter.



Installing the Remote Desktop Manager PowerShell module.

3. Import the PowerShell module via Import-Module, and verify that the module is now available for use with Get-Module as shown below. To view all available commands, use Get-Command -Module RemoteDesktopManager.

```
Import-Module RemoteDesktopManager
Get-Module RemoteDesktopManager
```



Importing the module and verifying that the module is available for use.

Adding a Data Source for DVLS with Windows Authentication

When you first launch RDM, a default SQLite local vault is created and available for use. In this article, a remote Devolutions Server (DVLS) connection is used to store and manage vault and entry permissions.

The PowerShell cmdlets operate on the RemoteDesktopManager.cfg file, typically stored in a location such as "$($Env:LOCALAPPDATA)\\Devolutions\\RemoteDesktopManager". If changes do not appear, you may need to verify that your settings are operating on the same file.

1. Since a Devolutions Server (DVLS) is in use, the first step is to add the DVLS data source. To verify this, use the Get-RDMDataSource cmdlet which lists all locally known RDM data sources. Here, the default SQLite “Local Data Source” is listed.

```
Get-RDMDataSource
```

```
PS C:\Users\testaccount1> Get-RDMDataSource

ID          : 4c30af75-054a-4aeb-8d1e-8a873c9038cb
IsConnected : False
IsOffline   : False
Name        : Local Data Source
Type        : SQLite
```

Listing all RDM Data Sources.

2. To add a new Data Source, you must first create the data source with the New-RDMData-Source cmdlet. Initially, the data source only exists in memory. The created object must then be passed to Set-RDMDataSource which applies the changes to your RDM instance.

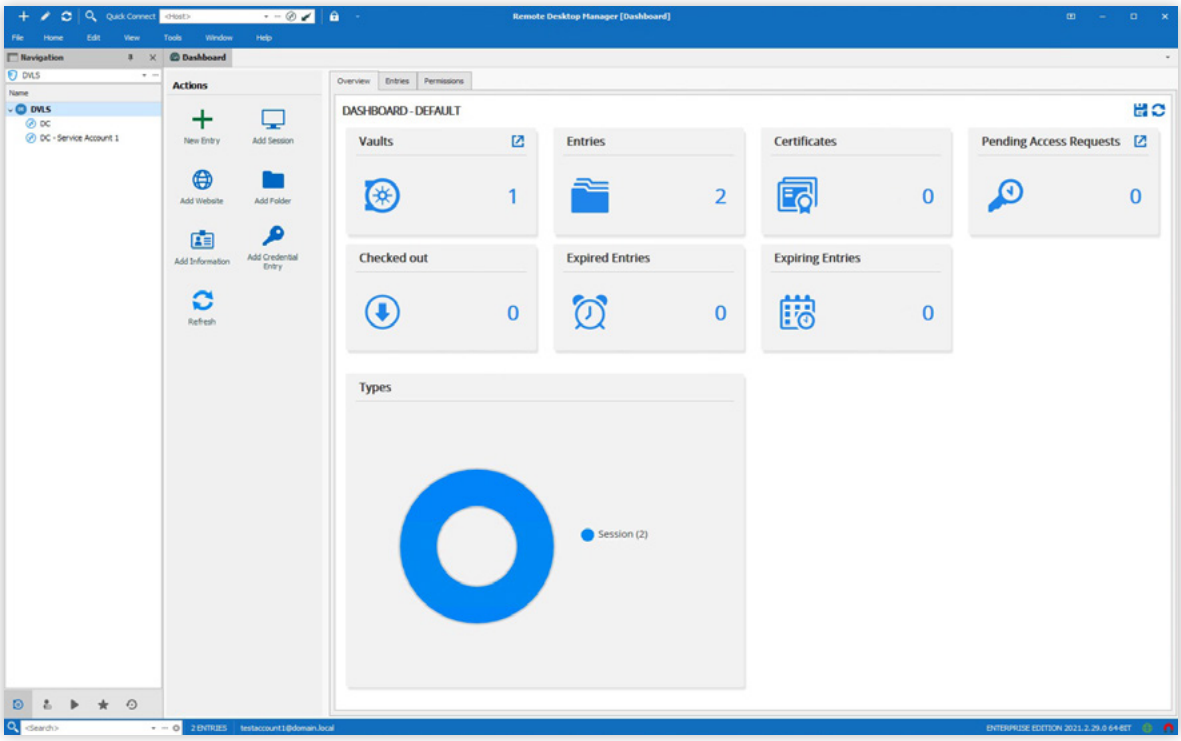
As this DVLS instance is connected to an Active Directory domain, for ease of operation, this data source uses Windows Authentication to transparently connect the currently logged-in user.

```
$DataSource = New-RDMDataSource -DVLS -Name ‘DVLS’ -Server ‘<https://devsrv.domain.local/dps>’
Set-RDMDataSourceProperty -DataSource $DataSource -Property ‘UseWindowsAuthentication’ -Value
$True
$DataSource | Set-RDMDataSource | Update-RDMUI
```

```
PS C:\Users\testaccount1> $DataSource = New-RDMDataSource -DVLS -Name 'DVLS' -Server 'https://devsrv.domain.local/dps'
PS C:\Users\testaccount1> Set-RDMDataSourceProperty -DataSource $DataSource -Property 'UseWindowsAuthentication' -Value
$True
PS C:\Users\testaccount1> $DataSource | Set-RDMDataSource | Update-RDMUI
PS C:\Users\testaccount1>
```

Creating the new DVLS data source with Windows Authentication.

3. Upon launch of RDM, you will have a new data source option to choose from, within the Navigation drop-down. Choose **DVLS** and due to the integrated Windows Authentication, the entries that this account has permission to are immediately available.



Connecting to the DVLS Data Source.

Adding a New Vault (Repository)

Up to this point, a single vault (repository) has been used, but to separate entries, it may be best to create a new vault. Although you can create a new local vault to store the entries, since DVLS is in use, it is best to create the new vault on the DVLS server for multiple users to access.

1. Before adding a new vault (repository) to your DVLS data source via PowerShell, you must first retrieve the RDM DVLS data source, named DVLS in this example. Next, set the current data source to the newly retrieved DVLS data source, and finally update the RDM UI to reflect your changes.

> *You must be an Administrator within DVLS to create a vault. If you have just changed your license type, exit the PowerShell session and re-open.*

```
$DataSource = Get-RDMDataSource -Name 'DVLS'
$DataSource | Set-RDMCurrentDataSource | Update-RDMUI
```

```
PS C:\Users\testaccount1> $DataSource = Get-RDMDataSource -Name 'DVLS'
PS C:\Users\testaccount1> $DataSource | Set-RDMCurrentDataSource | Update-RDMUI
PS C:\Users\testaccount1>
```

Changing to the correct DVLS data source.

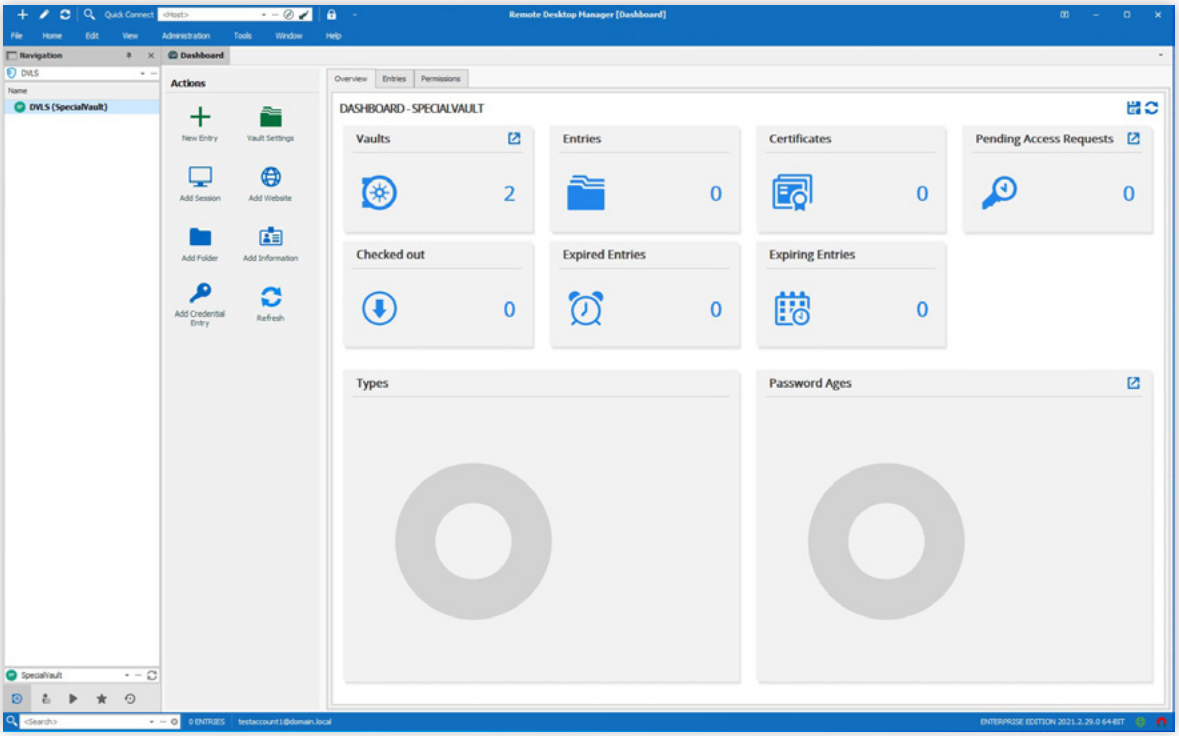
2. Next, create a new vault titled, SpecialVault in this example, with the New-RDMRepository cmdlet. A name is provided, along with the -SetRepository parameter as well, which takes the place of providing an extra command call to Set-RDMCurrentRepository.

```
$Vault = New-RDMRepository -Name 'SpecialVault' -SetRepository
Update-RDMUI
```

```
PS C:\Users\testaccount1> $Vault = New-RDMRepository -Name 'SpecialVault' -SetRepository
PS C:\Users\testaccount1> Update-RDMUI
PS C:\Users\testaccount1>
```

Creating a new vault.

3. Verify that the new vault has been created by launching RDM and navigating to the newly created SpecialVault as shown below by choosing it in the lower-right navigation drop-down.



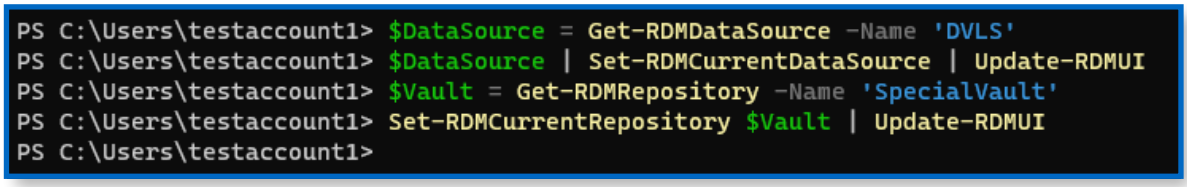
Navigating to the newly created vault.

Creating New Folders in SpecialVault

Now that the vault has been created, it helps to differentiate differing sessions (entries) into separate folders for logical separation and easier management.

1. As before, verify that you are operating on the correct data source and vault (repository), as shown in the code below.

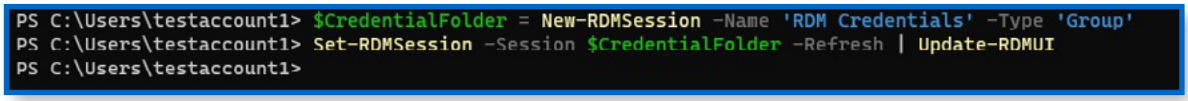
```
$DataSource = Get-RDMDataSource -Name 'DVLS'
$DataSource | Set-RDMCurrentDataSource | Update-RDMUI
$Vault = Get-RDMRepository -Name 'SpecialVault'
Set-RDMCurrentRepository $Vault | Update-RDMUI
```



Connecting to the correct data source and vault (repository).

2. Next, create a folder named RDM Credentials with the New-RDMSession cmdlet, set to the -Type of Group which denotes a folder type. Pass the newly created in-memory object to Set-RDMSession to apply the changes to RDM and DVLS.

```
$RDMCredentialFolder = New-RDMSession -Name 'RDM Credentials' -Type 'Group'
Set-RDMSession -Session $RDMCredentialFolder -Refresh | Update-RDMUI
```



Creating an RDM Credentials entry folder.

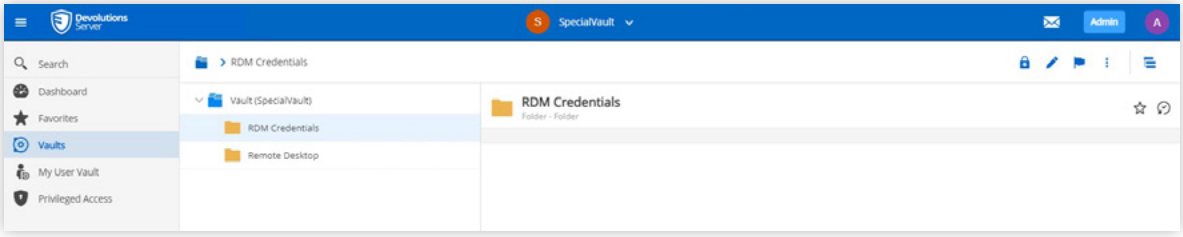
3. Once again, create a Remote Desktop folder with the same method.

```
$RemoteDesktopFolder = New-RDMSession -Name 'Remote Desktop' -Type 'Group'
Set-RDMSession -Session $RemoteDesktopFolder -Refresh | Update-RDMUI
```

```
PS C:\Users\testaccount1> $RemoteDesktopFolder = New-RDMSession -Name 'Remote Desktop' -Type 'Group'
PS C:\Users\testaccount1> Set-RDMSession -Session $RemoteDesktopFolder -Refresh | Update-RDMUI
PS C:\Users\testaccount1>
```

Creating a Remote Desktop entry folder.

4. As you can see below, there are now two folders created in the SpecialVault. This will be used to store Remote Desktop credentials and entries.



Newly created folders.

Creating a New Entry

With the newly created vault, you need a session (entry) to connect to. Follow the steps below to create a new session saved to the SpecialVault.

1. Verify that you are operating on the correct data source and vault (repository), as shown below.

```
$DataSource = Get-RDMDataSource -Name 'DVLS'
$DataSource | Set-RDMCurrentDataSource | Update-RDMUI
$Vault = Get-RDMRepository -Name 'SpecialVault'
Set-RDMCurrentRepository $Vault | Update-RDMUI
```

```
PS C:\Users\testaccount1> $DataSource = Get-RDMDataSource -Name 'DVLS'
PS C:\Users\testaccount1> $DataSource | Set-RDMCurrentDataSource | Update-RDMUI
PS C:\Users\testaccount1> $Vault = Get-RDMRepository -Name 'SpecialVault'
PS C:\Users\testaccount1> Set-RDMCurrentRepository $Vault | Update-RDMUI
PS C:\Users\testaccount1>
```

Connecting to the correct data source and vault (repository).

2. Next, set up the details of the session to create. First, create a credential session that will later be used as credentials for a specific RDP connection. This decouples the credential from an entry to make updating a credential used across multiple sessions easier.

```
$RDMCredential = New-RDMSession -Name "RDPConnection" -Type 'Credential' -Group
'RDM Credentials'
$RDMCredential.Credentials.UserName = "domain.local\\TestAccount1"
Set-RDMSession $RDMCredential -Refresh
Set-RDMSessionPassword -ID $RDMCredential.ID -Password (ConvertTo-SecureString
'_eUDMYQr7gP22eJz' -AsPlainText -Force) | Update-RDMUI
```

```
PS C:\Users\testaccount1> $RDMCredential = New-RDMSession -Name "RDPConnection" -Type 'Credential' -Group 'RDM Credentia
ls'
PS C:\Users\testaccount1> $RDMCredential.Credentials.UserName = "domain.local\TestAccount1"
PS C:\Users\testaccount1> Set-RDMSession $RDMCredential -Refresh
PS C:\Users\testaccount1> Set-RDMSessionPassword -ID $RDMCredential.ID -Password (ConvertTo-SecureString '_eUDMYQr7gP22e
Jz' -AsPlainText -Force) | Update-RDMUI
PS C:\Users\testaccount1>
```

Creating an RDP credential entry for use in later entries.

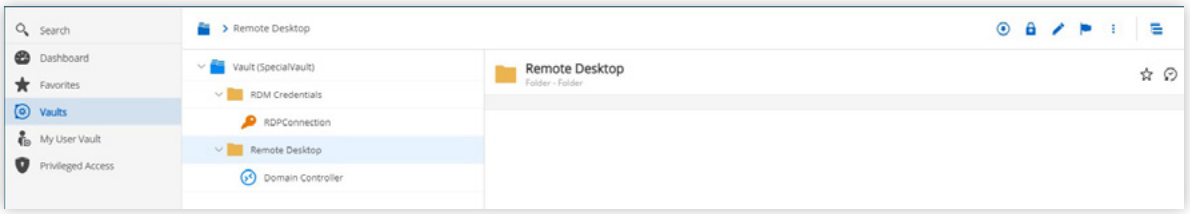
3. Now that the credential has been created, create a basic Remote Desktop session, as shown below, and assigned the previously created connection ID.

```
$RemoteDesktop = New-RDMSession -Name 'Domain Controller' -Type 'RDPConfigured'
-Group 'Remote Desktop'
$RemoteDesktop.Host = 'dc.domain.local'
$RemoteDesktop.CredentialConnectionID = $RDMCredential.ID
Set-RDMSession $RemoteDesktop -Refresh | Update-RDMUI
```

```
PS C:\Users\testaccount1> $RemoteDesktop = New-RDMSession -Name 'Domain Controller' -Type 'RDPConfigured' -Group 'Remote
 Desktop'
PS C:\Users\testaccount1> $RemoteDesktop.Host = 'dc.domain.local'
PS C:\Users\testaccount1> $RemoteDesktop.CredentialConnectionID = $RDMCredential.ID
PS C:\Users\testaccount1> Set-RDMSession $RemoteDesktop -Refresh | Update-RDMUI
PS C:\Users\testaccount1>
```

Creating the RDP entry itself for the actual connection.

4. Verify that the entries now exist in DVLS, by navigating to **Devolutions Server → Vaults → SpecialVault** and expand **RDM Credentials** and **Remote Desktop** folders.



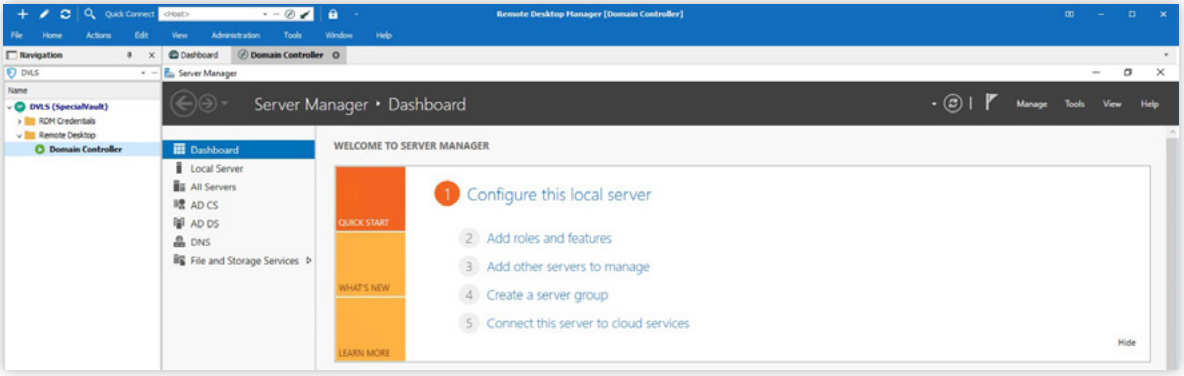
Verifying that the entries exist in the vault.

5. Connect to the session with the newly created entry.

```
Open-RDMSession -ID $RemoteDeskop.ID
```

```
PS C:\Users\testaccount1> Open-RDMSession -ID $RemoteDesktop.ID
PS C:\Users\testaccount1>
```

Opening a session from the command line with RDM.



Displaying the opened session in RDM.

If there is an error, this will silently fail.

Adding a New User

Perhaps you have onboarded a new user, but instead of navigating to the Devolutions Server interface, leverage PowerShell to create the new user in DVLS.

> *The user must already exist in Active Directory prior to adding to DVLS.*

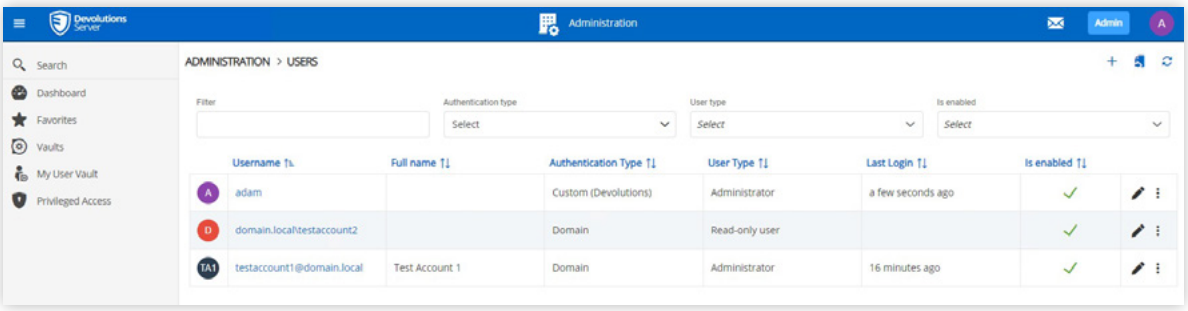
1. In this example, the New-RDMUser is passed an Active Directory domain account, set to the -AuthentificationType of Domain. Apply the changes with the Set-RDMUser cmdlet.

```
$User = New-RDMUser -Login 'domain.local\\testaccount2' -AuthentificationType 'Domain'
Set-RDMUser -User $User
```

```
PS C:\Users\testaccount1> $User = New-RDMUser -Login 'domain.local\testaccount2' -AuthentificationType 'Domain'
PS C:\Users\testaccount1> Set-RDMUser -User $User
PS C:\Users\testaccount1>
```

Creating a new DVLS user.

2. As shown in the DVLS web interface, the new user is shown below, added by default as a **Read-only user**.



Verifying that the new DVLS domain user was added.

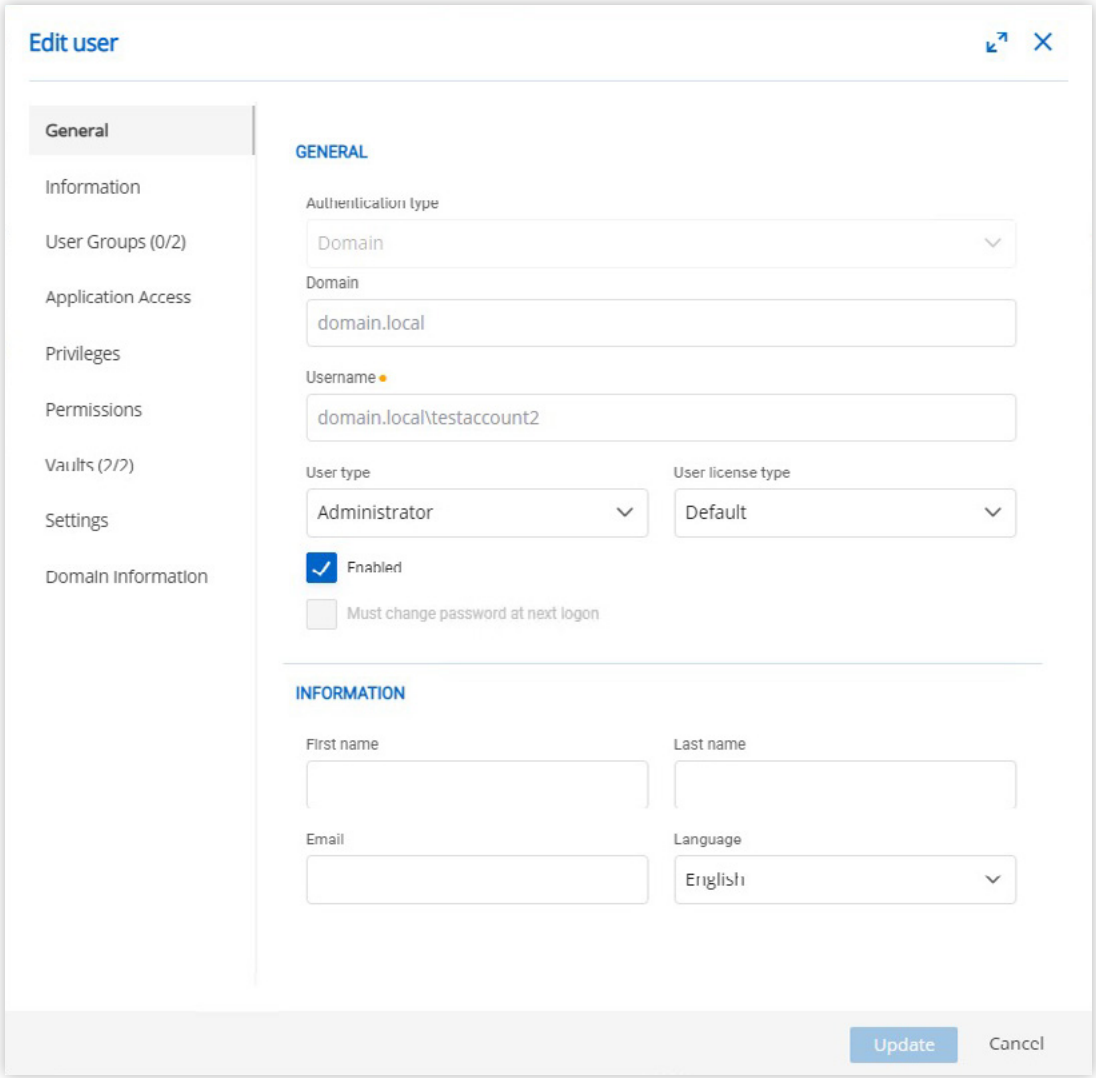
3. In some instances, you may want to specify an account as an Administrator. Here, you retrieve an existing user with Get-RDMUser to set IsAdministrator to $True. Apply the changes with Set-RDMUser.

```
$User = Get-RDMUser -Name 'domain.local\\testaccount2'
$User.IsAdministrator = $True
Set-RDMUser -User $User
```

```
PS C:\Users\testaccount1> $User = Get-RDMUser -Name 'domain.local\testaccount2'
PS C:\Users\testaccount1> $User.IsAdministrator = $True
PS C:\Users\testaccount1> Set-RDMUser -User $User
PS C:\Users\testaccount1>
```

Setting a user as an administrator.

4. Verify that the changes were made by navigating to the **Devolutions Server → Administration → Users** and clicking on the username. Under the **User Type** section, note that the type is now **Administrator**.



Verifying the user permission changes.

Adding a New Role

With the new user, perhaps you would like to assign the vault permissions to a specific group, add the new user to that Active Directory group, and remove the custom permissions added.

> *The group must already exist in Active Directory prior to adding to DVLS.*

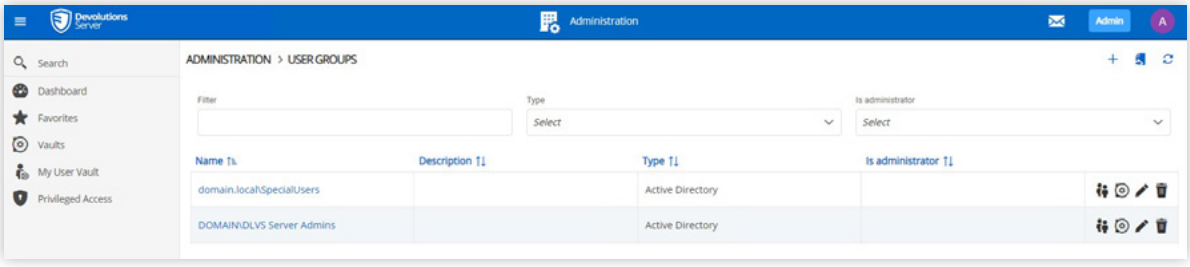
1. In the below example, first, you will create a new role with New-RDMRole that is backed by the existing Active Directory group domain.local\\SpecialUsers. Next, apply the changes with the Set-RDMRole cmdlet.

```
$Role = New-RDMRole -Name 'domain.local\\SpecialUsers'
Set-RDMRole -Role $Role
```

```
PS C:\Users\testaccount1> $Role = New-RDMRole -Name 'domain.local\SpecialUsers'
PS C:\Users\testaccount1> Set-RDMRole -Role $Role
PS C:\Users\testaccount1>
```

Creating the new RDM role.

2. Verify that the role now exists in **Devolutions Server → Administration → User Groups** as shown below.



Verifying the newly created role.

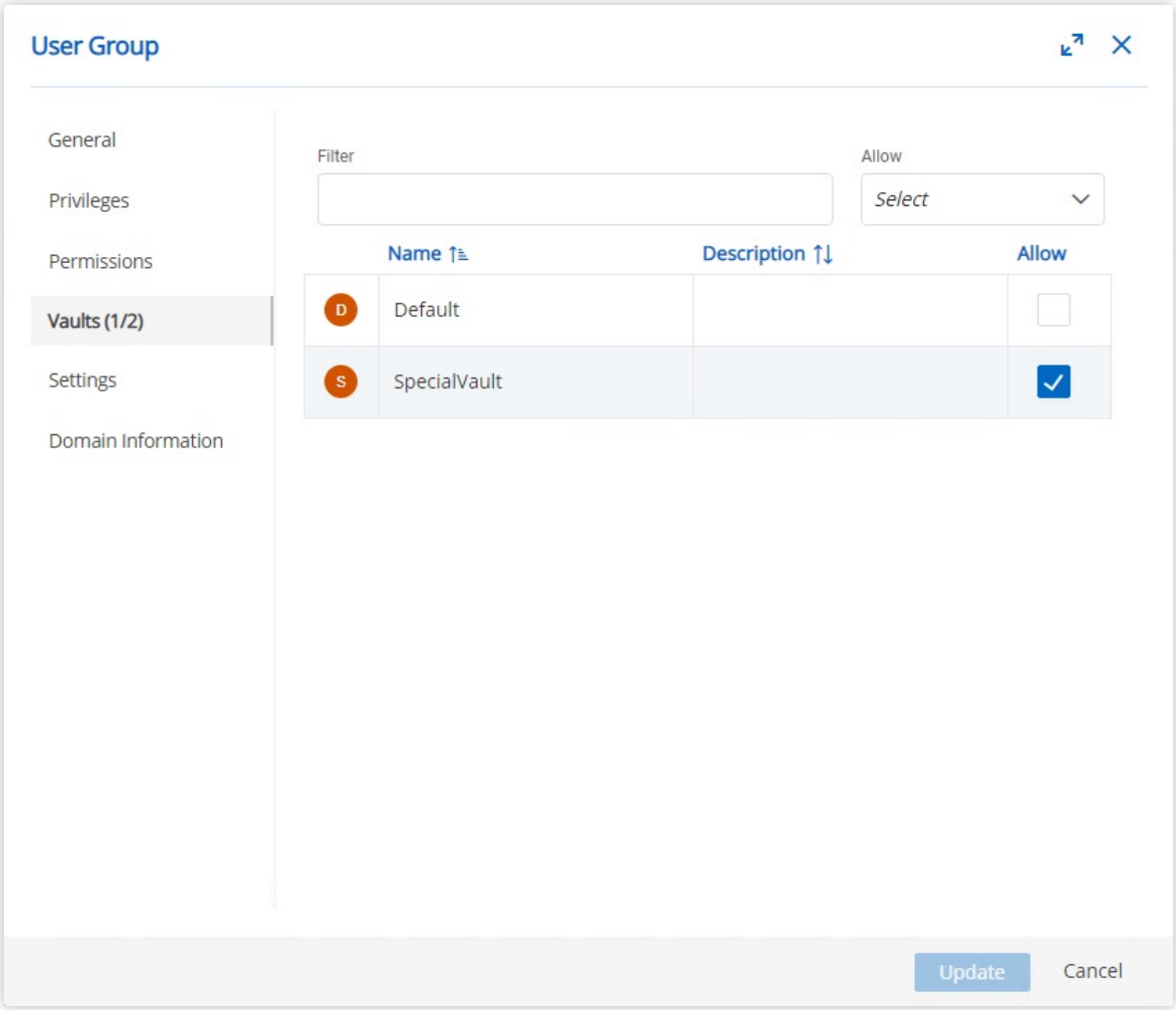
3. Next, you will allow the role, domain.local\\SpecialUsers, to have access to the SpecialVault.

```
$Vault = Get-RDMRepository -Name 'SpecialVault'
Set-RDMCurrentRepository -Repository $Vault | Update-RDMUI
Add-RDMRoleRepositoryAccess -Role $Role -Repository $Vault
Set-RDMRole -Role $Role
```

```
PS C:\Users\testaccount1> $Vault = Get-RDMRepository -Name 'SpecialVault'
PS C:\Users\testaccount1> Set-RDMCurrentRepository -Repository $Vault | Update-RDMUI
PS C:\Users\testaccount1> Add-RDMRoleRepositoryAccess -Role $Role -Repository $Vault
PS C:\Users\testaccount1> Set-RDMRole -Role $Role
PS C:\Users\testaccount1>
```

Granting the newly created role access to a vault.

4. Verify that the changes have been made by navigating to **Devolutions Server → Administration → User Groups → Vaults.**



Verifying that the role now has allow access.

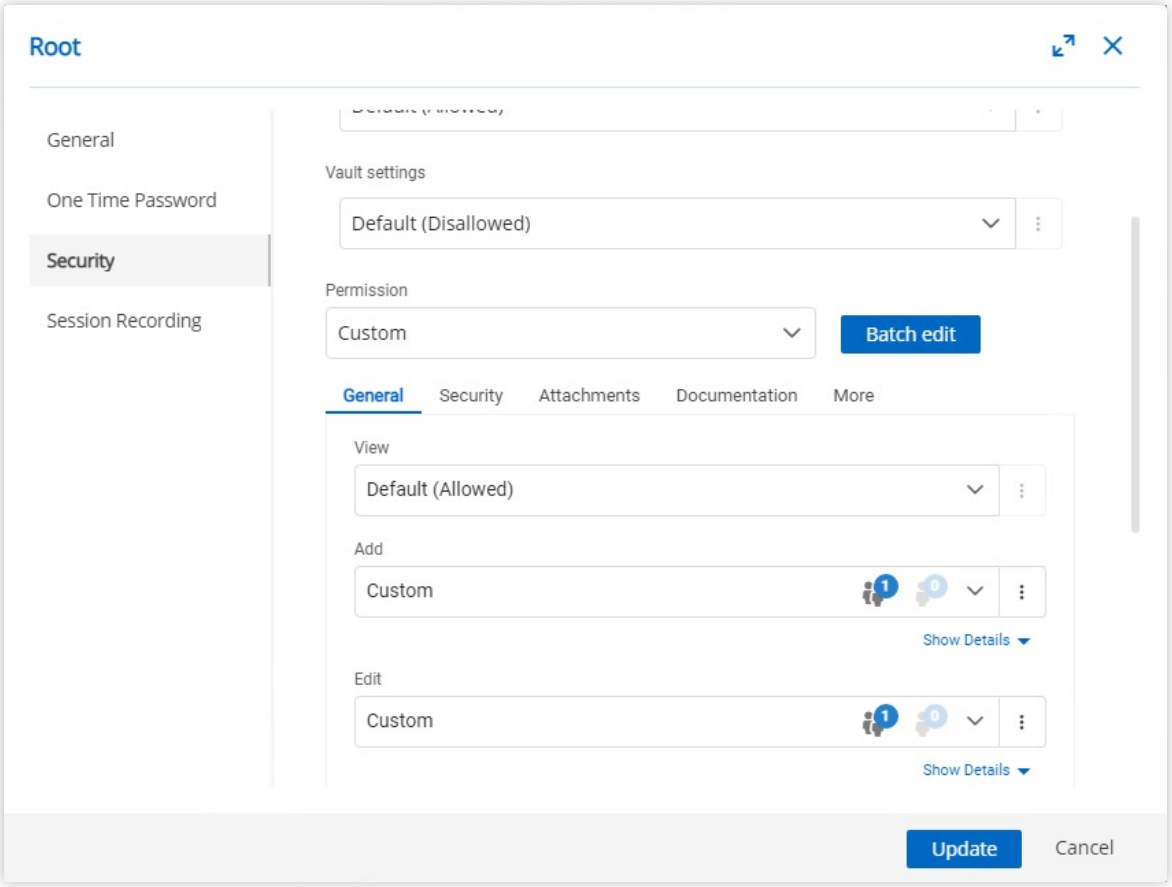
5. Modify the permissions for the role to allow for adding and editing of entries in the SpecialVault.

```
$Permissions = @(
              [Devolutions.RemoteDesktopManager.Business.ConnectionPermission]@{
                     'Override' = 'Custom'
              'Right' = 'Add'
              'Roles' = 'domain.local\\SpecialUsers'
              'RoleValues' = 'domain.local\\SpecialUsers'
              }
              [Devolutions.RemoteDesktopManager.Business.ConnectionPermission]@{
              'Override' = 'Custom'
              'Right' = 'Edit'
              'Roles' = 'domain.local\\SpecialUsers'
                       'RoleValues' = 'domain.local\\SpecialUsers'
              }
)

$RDMRoot = Get-RDMRootSession
$RDMRoot.Security.RoleOverride = 'Custom'
$RDMRoot.Security.Permissions = $Permissions

$RDMRoot | Set-RDMRootSession
```

```
PS C:\Users\testaccount1> $Permissions = @(
>>    [Devolutions.RemoteDesktopManager.Business.ConnectionPermission]@{
>>      'Override' = 'Custom'
>>      'Right' = 'Add'
>>      'Roles' = 'domain.local\SpecialUsers'
>>      'RoleValues' = 'domain.local\SpecialUsers'
>>    }
>>    [Devolutions.RemoteDesktopManager.Business.ConnectionPermission]@{
>>      'Override' = 'Custom'
>>      'Right' = 'Edit'
>>      'Roles' = 'domain.local\SpecialUsers'
>>      'RoleValues' = 'domain.local\SpecialUsers'
>>    }
>> )
PS C:\Users\testaccount1> $RDMRoot = Get-RDMRootSession
PS C:\Users\testaccount1> $RDMRoot.Security.RoleOverride = 'Custom'
PS C:\Users\testaccount1> $RDMRoot.Security.Permissions = $Permissions
PS C:\Users\testaccount1> $RDMRoot | Set-RDMRootSession
PS C:\Users\testaccount1>
```

Modifying permissions for the role.

6. Verify that the changes have been made by navigating to **Devolutions Server → Vaults → Properties (right-click on the root of the vault) → Security → Permissions.**



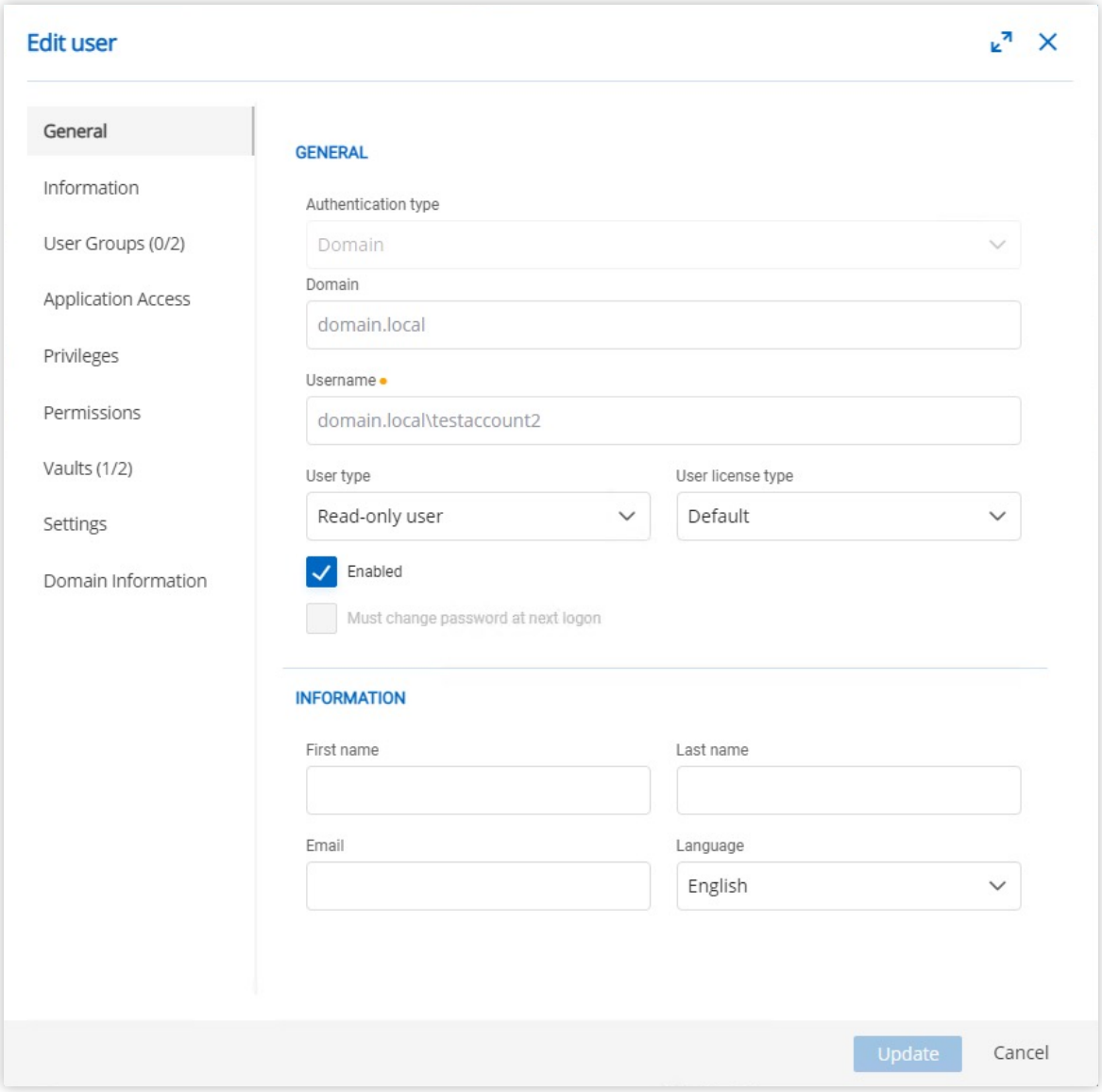
Verifying that the role has custom permissions in the vault.

7. Finally, remove the custom user permissions, previously added, as the user now has these same permissions due to being a member of the Active Directory group.

```
$User = Get-RDMUser -Name 'domain.local\\testaccount2'
$User.CanAdd = $False
$User.CanEdit = $False
Set-RDMUser -User $User
```

```
PS C:\Users\testaccount1> $User = Get-RDMUser -Name 'domain.local\testaccount2'
PS C:\Users\testaccount1> $User.CanAdd = $False
PS C:\Users\testaccount1> $User.CanEdit = $False
PS C:\Users\testaccount1> Set-RDMUser -User $User
PS C:\Users\testaccount1>
```

Remove the custom Add and Edit permissions from the user.



Verify that the user no longer has custom permissions.

Exporting and Importing Sessions for Backup

How would you go about creating a backup of your sessions in the event of a disaster? With the Export-RDMSession cmdlet, an XML file containing all the necessary data is easy to export.

1. Verify that you are operating on the correct data source and vault (repository), as shown below.

```
$DataSource = Get-RDMDataSource -Name 'DVLS'
$DataSource | Set-RDMCurrentDataSource | Update-RDMUI
$Vault = Get-RDMRepository -Name 'SpecialVault'
Set-RDMCurrentRepository $Vault | Update-RDMUI
```

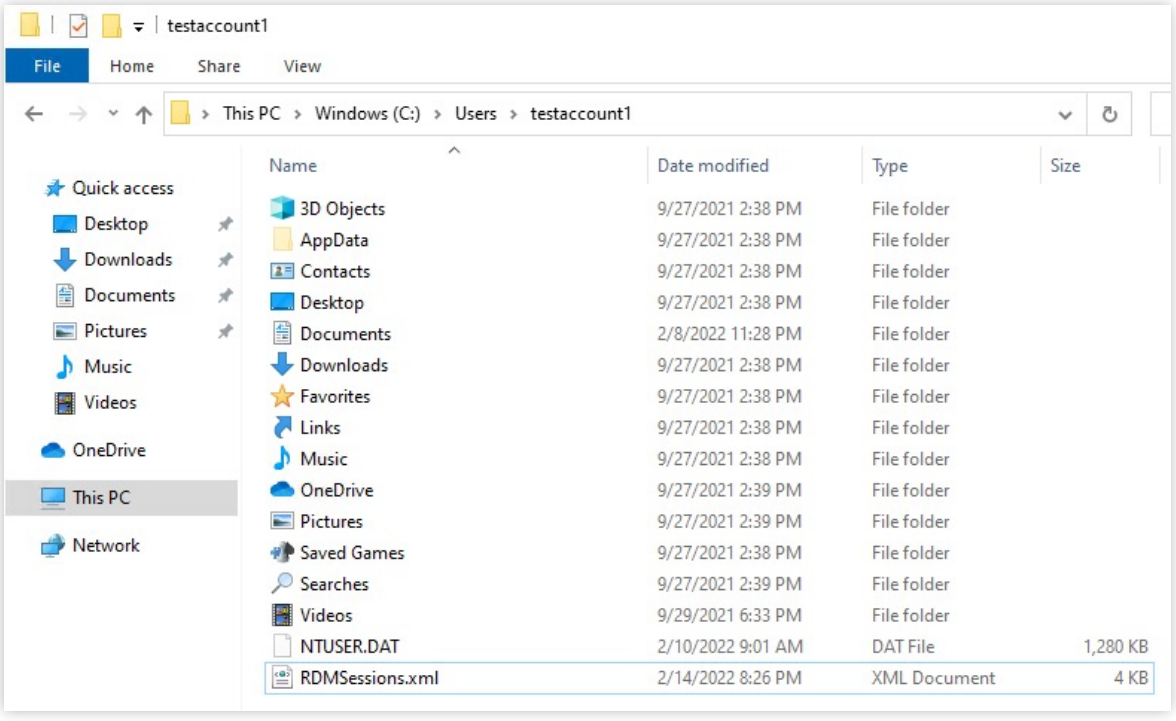
2. Next, retrieve all sessions and then export the XML file to the given path which, in this example, includes both credentials and security groups.

```
$Sessions = Get-RDMSession
Export-RDMSession -Sessions $Sessions -Path "$($Env:USERPROFILE)\\RDMSessions.xml" -XML
-IncludeCredentials -IncludeSecurityGroups
```

```
PS C:\Users\testaccount1> $Sessions = Get-RDMSession
PS C:\Users\testaccount1> Export-RDMSession -Sessions $Sessions -Path "$($Env:USERPROFILE)\RDMSessions.xml" -XML -Includ
eCredentials -IncludeSecurityGroups
PS C:\Users\testaccount1>
```

Exporting the XML sessions data.

3. Verify that the exported file exists as expected, as shown by the file RDMSessions.xml seen below.



Verifying the RDMSessions.xml file exists.

4. Finally, add the exported sessions into DVLS with the Import-RDMSession cmdlet.

```
Import-RDMSession -Path "$($Env:USERPROFILE)\\RDMSessions.xml"
```

```
PS C:\Users\testaccount1> Import-RDMSession -Path "$($Env:USERPROFILE)\RDMSessions.xml"


AlternateShell                          :
AlwaysAskForResources                   : False
ApplicationIntegrationMode              : Default
AuthentificationLevel                   : Default
AutoReconnection                        : True
CommandLine                             :
CommandLineWaitForApplicationToExit     : False
CommandLineWorkingDirectory             :
Console                                 : False
DesktopComposition                      : False
DisableBitmapCache                      : False
DisableCursorSetting                    : False
DisableFullWindowDrag                   : False
DisableMenuAnims                        : False
DisableThemes                           : False
DisableWallpaper                        : False
```

Import RDM Sessions into DVLS.

Exporting and Importing Roles for Backup

How would you go about creating a backup of your roles and their permissions? Although there is not a standardized cmdlet to do this, you can utilize built-in PowerShell capabilities to export the data to an XML file for later import.

1. Verify that you are operating on the correct data source and vault (repository), as shown below.

```
$DataSource = Get-RDMDataSource -Name 'DVLS'
$DataSource | Set-RDMCurrentDataSource | Update-RDMUI
$Vault = Get-RDMRepository -Name 'SpecialVault'
Set-RDMCurrentRepository $Vault | Update-RDMUI
```

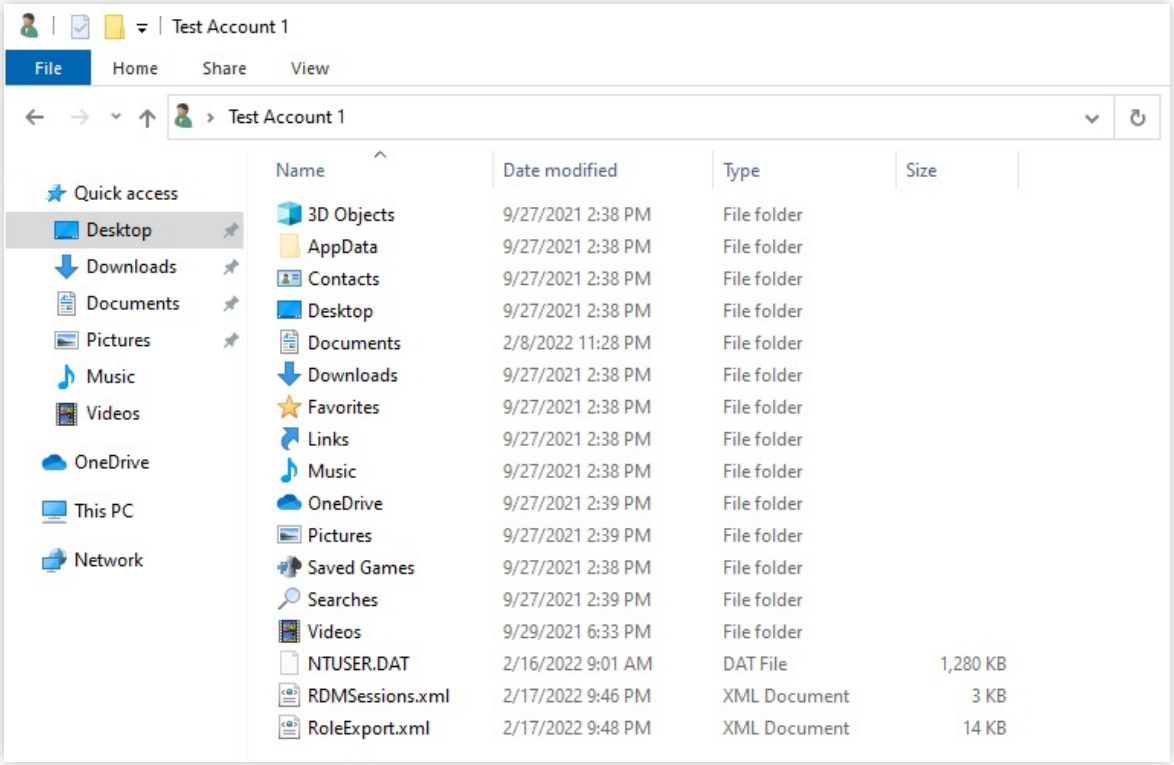
2. Next, retrieve all roles and then export the XML file to the given path shown below.

```
$Roles = Get-RDMRole
$Roles | Export-Clixml -Path "$($Env:USERPROFILE)\\RoleExport.xml"
```

```
PS C:\Users\testaccount1> $Roles = Get-RDMRole
PS C:\Users\testaccount1> $Roles | Export-Clixml -Path "$($Env:USERPROFILE)\RoleExport.xml"
PS C:\Users\testaccount1>
```

Exporting RDM roles.

3. Confirm that the roles have been exported as seen by the RoleExport.xml file shown below.



Verifying the exported roles file exist.

4. A simple script that takes the exported role data and subsequently imports the roles into DVLS is shown below.

```
Import-Clixml -Path "$($Env:USERPROFILE)\\RoleExport.xml" | ForEach-Object {
       $Role = New-RDMRole -Name $PSItem.Name
Set-RDMRole -Role $Role
Set-RDMRoleProperty -Property "CanAdd" -Role $Role -Value $PSItem.CanAdd
Set-RDMRoleProperty -Property "CanDelete" -Role $Role -Value $PSItem.CanDelete
Set-RDMRoleProperty -Property "CanEdit" -Role $Role -Value $PSItem.CanEdit
Set-RDMRoleProperty -Property "CustomSecurity" -Role $Role -Value $PSItem.CustomSecurity
Set-RDMRoleProperty -Property "Description" -Role $Role -Value $PSItem.Description
Set-RDMRoleProperty -Property "Email" -Role $Role -Value $PSItem.Email

If ($PSItem.IsAdministrator) {
Set-RDMRoleProperty -Property "IsAdministrator" -Role $Role -Value $True
}

             Set-RDMRole -Role $Role
}
```

```
PS C:\Users\testaccount1> Import-Clixml -Path "$($Env:USERPROFILE)\RoleExport.xml" | ForEach-Object {
>>     $Role = New-RDMRole -Name $PSItem.Name
>>
>>     Set-RDMRole -Role $Role
>>
>>     Set-RDMRoleProperty -Property "CanAdd" -Role $Role -Value $PSItem.CanAdd
>>     Set-RDMRoleProperty -Property "CanDelete" -Role $Role -Value $PSItem.CanDelete
>>     Set-RDMRoleProperty -Property "CanEdit" -Role $Role -Value $PSItem.CanEdit
>>     Set-RDMRoleProperty -Property "CustomSecurity" -Role $Role -Value $PSItem.CustomSecurity
>>     Set-RDMRoleProperty -Property "Description" -Role $Role -Value $PSItem.Description
>>     Set-RDMRoleProperty -Property "Email" -Role $Role -Value $PSItem.Email
>>
>>     If ($PSItem.IsAdministrator) {
>>         Set-RDMRoleProperty -Property "IsAdministrator" -Role $Role -Value $True
>>     }
>>
>>     Set-RDMRole -Role $Role
>> }
PS C:\Users\testaccount1>
```

Importing exported roles.

If the role already exists, you will get a WARNING: Unable to save user error. Additionally, you will need to set the role up again on any vault that previously had custom permissions set.