HOW TO

Leverage PowerShell Secret Management with Password Hub and Devolutions Server

Devolutions | WHERE IT MEETS SECURITY

Migrating to a new password management system can be difficult, but this process is made easier with the PowerShell Secret Management module for Password Hub and Devolutions Server. This extends to other password managers that have similar modules, as this standardized module makes it simple to replace existing systems with a secure and integrated password management solution.

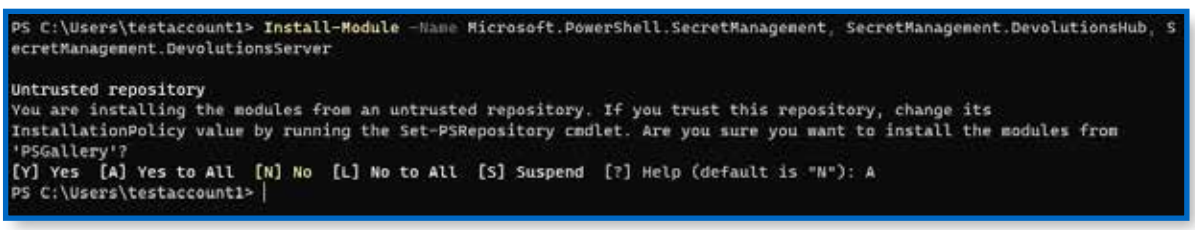
Installing the Devolutions Server and Hub PowerShell Secret Store Modules

Before you utilize the Devolutions Secret Management articles, you must first install the Microsoft Secret Management PowerShell module, along with Devolutions Hub and Devolutions Server secret store modules, all from the PowerShell Gallery.

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

2. Install the latest version of Microsoft.PowerShell.SecretManagement, SecretManagement.DevolutionsHub, and SecretManagement.DevolutionsServer modules, which at the time of this writing is v1.1.2, v0.3.0, and v0.2 respectively. 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 Microsoft.PowerShell.SecretManagement,
SecretManagement.DevolutionsHub, SecretManagement.DevolutionsServer
```

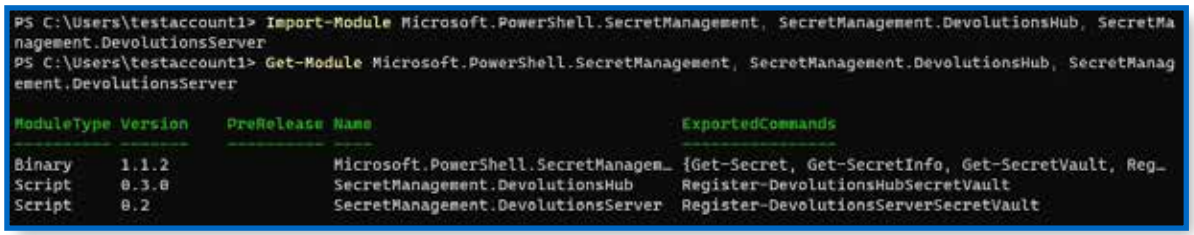


Installing the necessary modules

If you do not intend to use either Devolutions Hub or Devolutions Server, you may omit the installation of the module.

3. Import the PowerShell modules via Import-Module, and verify that the modules are now available for use with Get-Module as shown below.

```
Import-Module Microsoft.PowerShell.SecretManagement,
SecretManagement.DevolutionsHub, SecretManagement.DevolutionsServer
Get-Module Microsoft.PowerShell.SecretManagement, SecretManagement.DevolutionsHub,
SecretManagement.DevolutionsServer
```



Verifying that the modules installed correctly

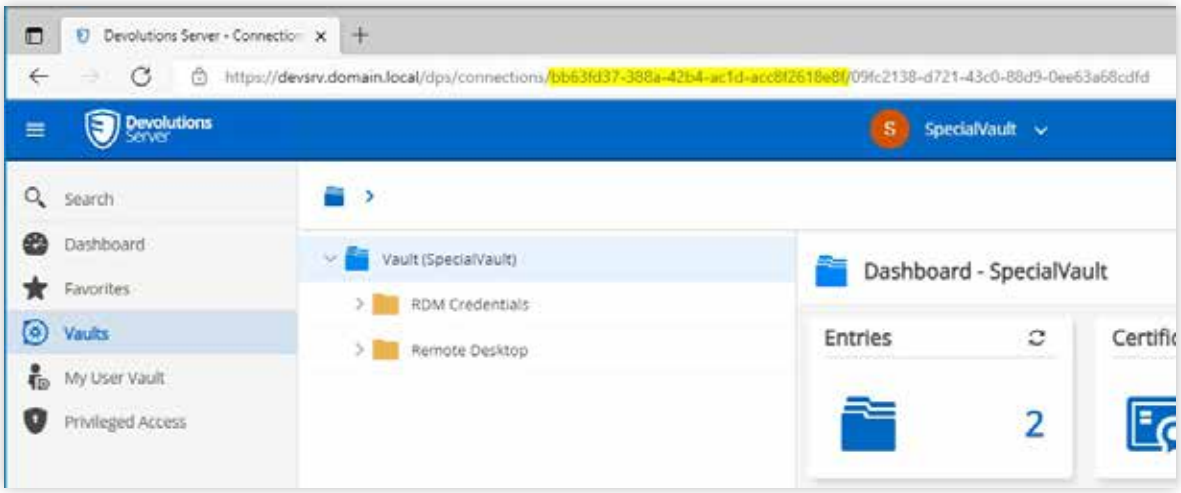
Configuring the Secret Store Modules

Before the modules may be used, you must first configure the modules to connect to their respective data stores. Register the relevant secret vaults to the Microsoft Secret Management module as shown below for either the Devolutions Server or Devolutions Hub.

This must only be done once, as a registered vault persists in future PowerShell sessions once the modules are imported.

Connecting Devolutions Server

1. To retrieve the Vault ID, locate the vault in the Devolutions Server web interface by navigating to **Devolutions Server → Vaults** and locating the ID in the URL itself as highlighted below.



Retrieving the Vault ID

2. Next, run the following code to register a secret vault for the Devolutions Server, providing the server URL and credentials with access to the provided vault ID.

```
$Credentials = Get-Credential

Register-SecretVault -Name 'DevolutionsServer' -ModuleName 'SecretManagement.
DevolutionsServer' -VaultParameters @{
    'Url'      = 'https://devsrv.domain.local/dps'
    'UserName' = $Credentials.UserName
    'Password' = ($Credentials.Password | ConvertFrom-SecureString -AsPlainText)
    'VaultId'  = 'bb63fd37-388a-42b4-ac1d-acc8f2618e8f'
}
```

```
PS C:\Users\testaccount1> $Credentials = Get-Credential

PowerShell credential request
Enter your credentials.
User: domain.local\testaccount1
Password for user domain.local\testaccount1: ****************

PS C:\Users\testaccount1> Register-SecretVault -Name 'DevolutionsServer' -ModuleName 'SecretManagement.DevolutionsServer
' -VaultParameters @{
>>     'Url'      = 'https://devsrv.domain.local/dps'
>>     'UserName' = $Credentials.UserName
>>     'Password' = ($Credentials.Password | ConvertFrom-SecureString -AsPlainText)
>>     'VaultId'  = 'bb63fd37-388a-42b4-ac1d-acc8f2618e8f'
>> }
PS C:\Users\testaccount1>
```

Registering the Secret Vault

If you are using the Remote Desktop Manager PowerShell module, you may use Get-RDMRepository to retrieve the required VaultId property which is shown as the returned Id property from the cmdlet.

3. Verify that the vault works as expected with the Test-SecretVault cmdlet as shown below. If True is returned, then the vault was successfully connected.

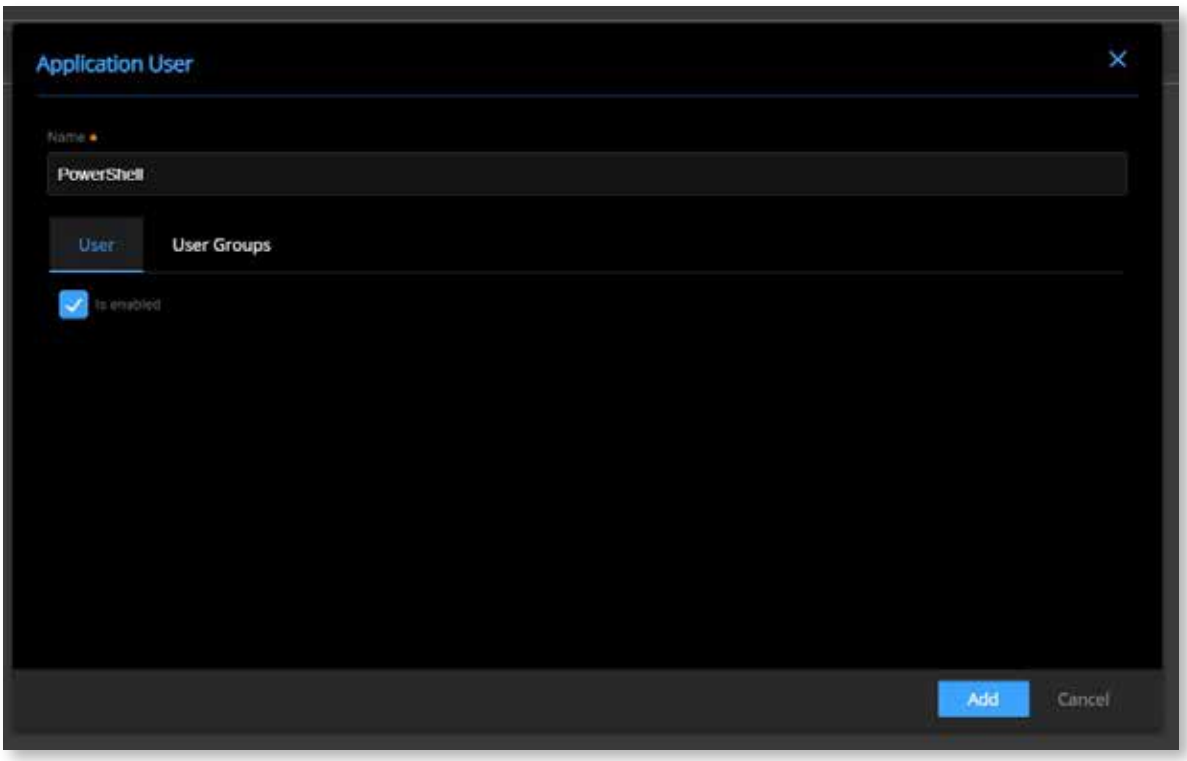
```
Test-SecretVault -Name 'DevolutionsServer'
```

```
PS C:\Users\testaccount1> Test-SecretVault -Name 'DevolutionsServer'
True
```

Verifying that the vault may connect

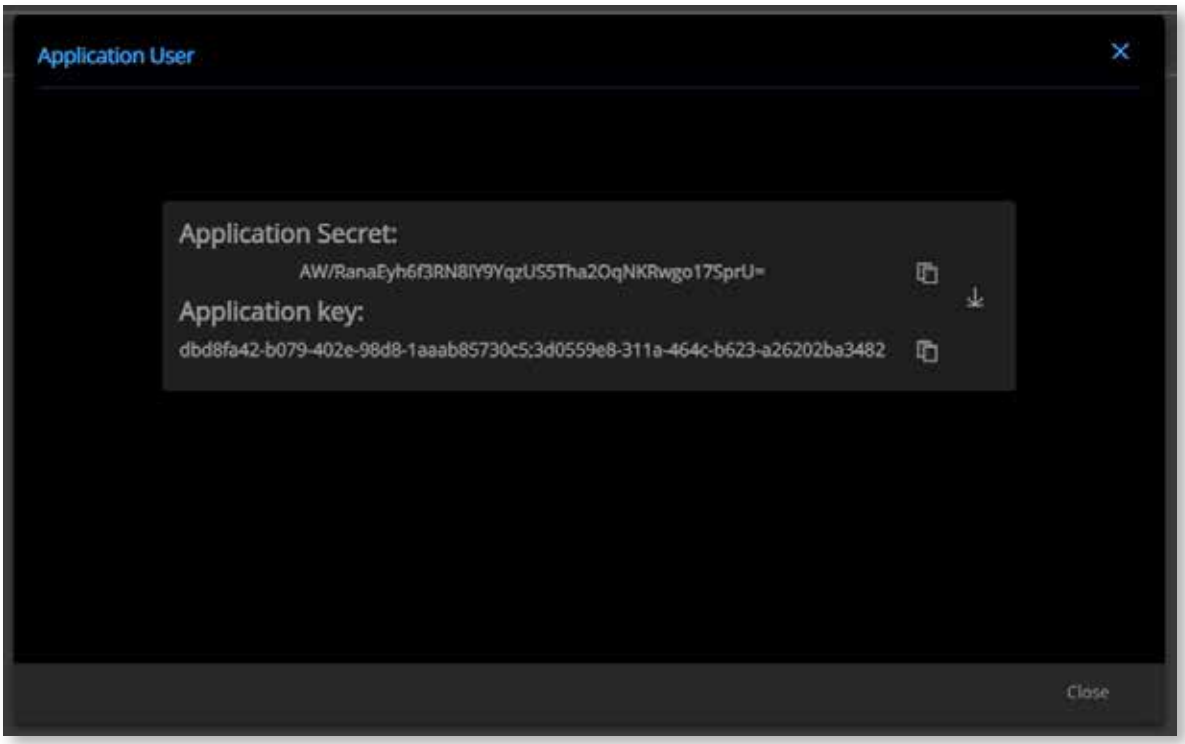
Connecting Devolutions Hub

1. First, you must create an Application User. Log in to the Devolutions Hub web interface and navigate to **Administration → Application Users**. Once there, click the **plus** button to add a new Application User providing a name and verifying that the **Is Enabled** option is checked.



Creating an Application User

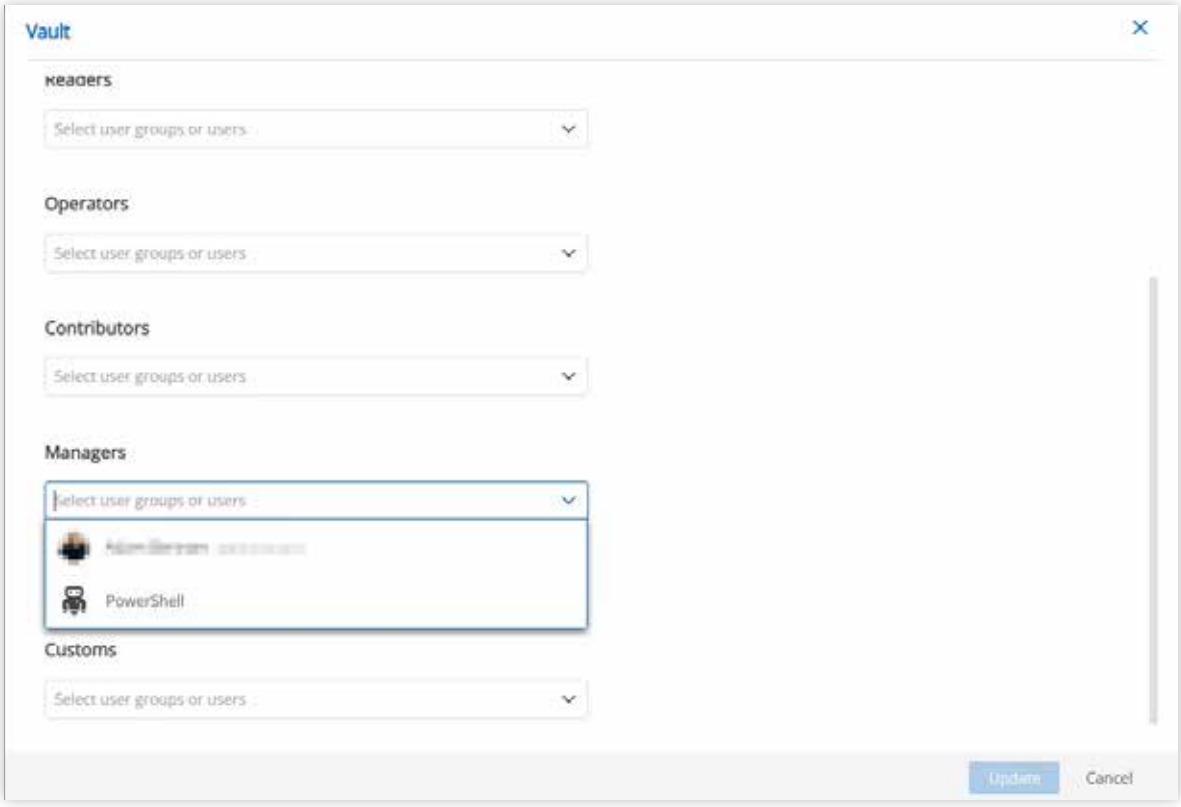
Copy the resulting Application Secret and Application Key, as they will be used to connect the Secret Management module.



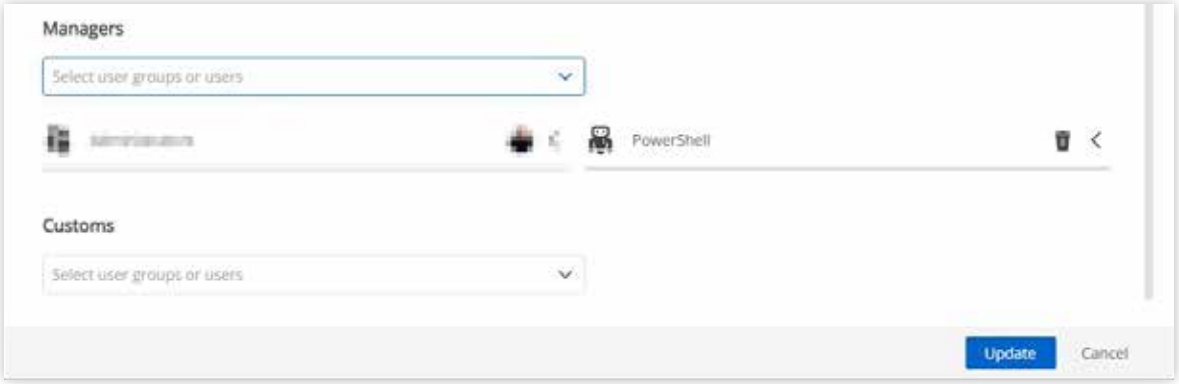
Retrieving the Application Secret and Key

> *The Application Secret will only be shown once, so make sure to download or copy this before clicking off the screen.*

2. Before the Application User can access a vault, you must first grant the account access. Navigate to **Administration → Vaults → Default Vault** (used in this tutorial) and click the **Edit** icon (pencil). For the purposes of this tutorial, the account is added as a Manager.

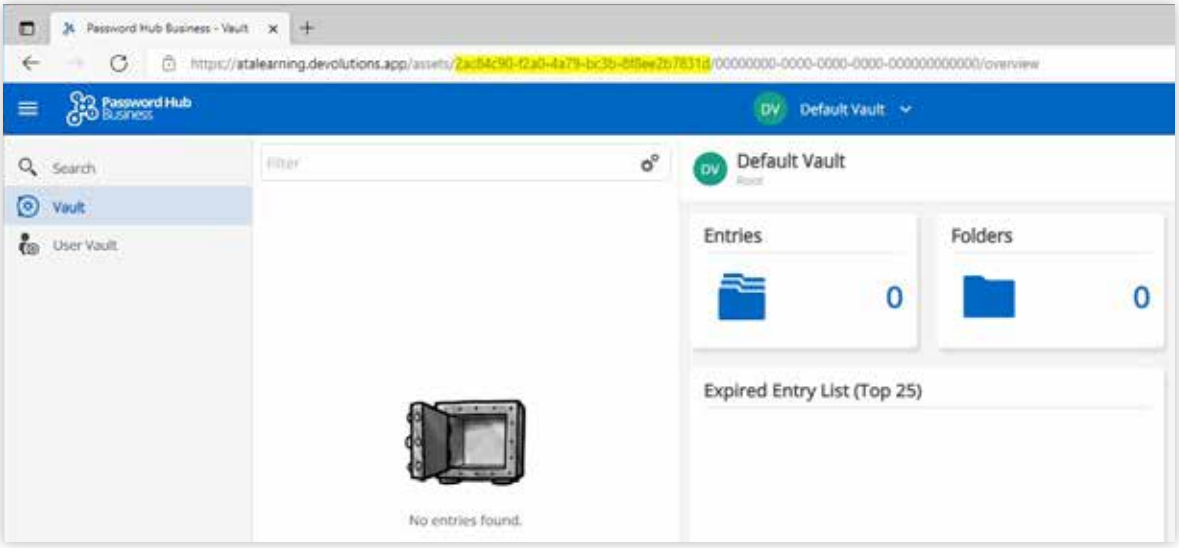


Selecting the Application User



Saving the Updated Permissions

3. To retrieve the Vault ID, locate the vault in the Devolutions Hub web interface by navigating to **Devolutions Hub → Vaults** and locating the ID in the URL itself as highlighted below.



Retrieving Vault ID

4. Next, run the following code to register a secret vault for the Devolutions Hub, providing the server URL and credentials with access to the provided vault ID.

```
Register-SecretVault -Name 'DevolutionsHub' -ModuleName 'SecretManagement.DevolutionsHub'
-VaultParameters @{
    'Url'               = 'https://atalearning.devolutions.app'
    'ApplicationKey'    = 'dbd8-
fa42-b079-402e-98d8-1aaab85730c5;3d0559e8-311a-464c-b623-a26202ba3482'
    'ApplicationSecret' = 'AW/RanaEyh6f3RN8IY9YqzUS5Tha2OqNKRwgo17SprU='
    'VaultId'           = '2ac84c90-f2a0-4a79-bc3b-8f8ee2b7831d'
}
```

```
PS C:\Users\testaccount1> Register-SecretVault -Name 'DevolutionsHub' -ModuleName 'SecretManagement.DevolutionsHub' -Vau
ltParameters @{
>>     'Url'               = 'https://atalearning.devolutions.app'
>>     'ApplicationKey'    = 'dbd8fa42-b079-402e-98d8-1aaab85730c5;3d0559e8-311a-464c-b623-a26202ba3482'
>>     'ApplicationSecret' = 'AW/RanaEyh6f3RN8IY9YqzUS5Tha2OqNKRwgo17SprU='
>>     'VaultId'           = '2ac84c90-f2a0-4a79-bc3b-8f8ee2b7831d'
>> }
PS C:\Users\testaccount1>
```

Registering Secret Vault

5. Verify that the vault works as expected with the Test-SecretVault cmdlet as shown below.

```
Test-SecretVault -Name 'DevolutionsHub'
```

```
PS C:\Users\testaccount1> Test-SecretVault -Name 'DevolutionsHub'
True
PS C:\Users\testaccount1>
```

Verifying that the vault may connect

Listing Vault Entries

To list the available entries from a vault, the Get-SecretInfo cmdlet will display information on returned entries. It is best to provide the -Vault parameter, especially if you multiple multiple vaults registered. As shown below, two entries are returned.

```
Get-SecretInfo -Vault 'DevolutionsServer'
```

The vault referenced by Get-SecretInfo is not the same as the vault within Devolutions Server or Hub. Here, the vault is either DevolutionsServer or DevolutionsHub.

```
PS C:\Users\testaccount1> Get-SecretInfo -Vault 'DevolutionsServer'
WARNING: [Close-DSSession] No session was previously established.

Name                                Type         VaultName
----                                ----         ---------
RDM Credentials\RDPConnection       PSCredential DevolutionsServer
Remote Desktop\Domain Controller PSCredential DevolutionsServer

PS C:\Users\testaccount1>
```

Retrieving available entries from the Devolutions Server

To list entries from the DevolutionsHub connection, change the specified vault to the Devolutions Hub connection as shown below.

```
PS C:\Users\testaccount1> Get-SecretInfo -Vault 'DevolutionsHub'

Name Type         VaultName
---- ----         ---------
DC   PSCredential DevolutionsHub
```

Retrieving the Devolutions Hub entries

Retrieving Vault Entries

To retrieve an individual vault entry, use the Get-Secret cmdlet, as demonstrated below with the Devolutions Server. You may notice that the returned value is a PSCredential object, the preferred method, but you may specify the AsPlainText option to retrieve the value unencrypted.

If there is no username or password property set, then the returned values will be blank.

```
Get-Secret -Name 'RDM Credentials\\RDPConnection' -Vault 'DevolutionsServer'
```

```
PS C:\Users\testaccount1> Get-Secret -Name 'RDM Credentials\RDPConnection' -Vault 'DevolutionsServer'
WARNING: [Close-DSSession] No session was previously established.

UserName                                  Password
--------                                  --------
domain.local\TestAccount1 System.Security.SecureString
```

Retrieve a specific entry from Devolutions Server

As with the Devolutions Server, to retrieve a secret from the Devolution Hub, use the same Get-Secret command, but specify DevolutionsHub as the vault.

```
Get-Secret -Name 'TestUser' -Vault 'DevolutionsHub'
```

```
PS C:\Users\testaccount1> Get-Secret -Name 'TestUser' -Vault 'DevolutionsHub'

UserName                  Password
--------                  --------
TestUser System.Security.SecureString
```

Retrieve a specific entry from Devolutions Hub

You may also pass in the GUID of the entry, instead of the name, to the same Name parameter to retrieve an entry. This method is typically faster.

Retrieving Vault Entries Passwords as Plaintext

Although the AsPlainText password property on Get-Secret does not return an unencrypted password, you may use the following command to decrypt the Password value. This works with retrieved entries from both Devolutions Server and Devolutions Hub.

```
Get-Secret -Name 'RDM Credentials\\RDPConnection' -Vault 'DevolutionsServer' | Select-Object
-ExpandProperty Password | ConvertFrom-SecureString -AsPlainText
```

```
PS C:\Users\testaccount1> Get-Secret -Name 'RDM Credentials\RDPConnection' -Vault 'DevolutionsServer' | Select-Object -E
xpandProperty Password | ConvertFrom-SecureString -AsPlainText
WARNING: [Close-DSSession] No session was previously established.
_eUDMYQr7gP22eJz
```

Retrieve and convert a password from an entry to plain text

Creating Vault Entries

To create a new secret in Devolutions Server, the Set-Secret cmdlet handles the creation of new secrets. As of this article, Set-Secret does not support the updating of existing secrets. A new entry will be created each time if an existing entry name is supplied.

```
$Credentials = Get-Credential
Set-Secret -Name 'ServiceUser' -Secret $Credentials -Vault 'DevolutionsServer'
```

```
PS C:\Users\testaccount1> $Credentials = Get-Credential

PowerShell credential request
Enter your credentials.
User: domain.local\testaccount1
Password for user domain.local\testaccount1: ****************

PS C:\Users\testaccount1> Set-Secret -Name 'ServiceUser' -Secret $Credentials -Vault 'DevolutionsServer'
WARNING: [Close-DSSession] No session was previously established.
```

Create a new entry in Devolutions Server

As of now, the Set-Secret functionality of Devolutions Hub does not work.

Incorporating Devolutions Secret Management into Scripts

Often, a server-side script requires credentials to manage a process. Instead of hard-coding that credential into the script, utilize the Devolutions Secret Management abilities to retrieve a credential for use, as shown in the simple example below.

```
# Verify that the vault connection is available
If (Test-SecretVault -Name 'DevolutionsServer') {
        $Credential = Get-Secret -Name 'ServiceUser' -Vault 'DevolutionsServer'

        # Verify that a credential is returned
        If ($Credential) {
                # Retrieve the service details
                $Service = Get-Service -Name "ImportantService"

                # If the existing service username does not match that of the credential
update the service
                If ($Service.UserName -NE $Credential.UserName) {
                        Write-Host "Updating Service Credential"
                        Set-Service -Name "ImportantService" -Credential $Credential
                }
        }
}
```

```
PS C:\Users\testaccount1> Import-Module Microsoft.PowerShell.SecretManagement, SecretManagement.DevolutionsServer
PS C:\Users\testaccount1>
PS C:\Users\testaccount1> If (Test-SecretVault -Name 'DevolutionsServer') {
>> $Credential = Get-Secret -Name 'ServiceUser' -Vault 'DevolutionsServer'
>>
>> If ($Credential) {
>> $Service = Get-Service -Name "ImportantService"
>>
>> If ($Service.UserName -NE $Credential.UserName) {
>> Write-Host "Updating Service Credential"
>> Set-Service -Name "ImportantService" -Credential $Credential
>> }
>> }
>> }
WARNING: [Close-DSSession] No session was previously established.
WARNING: [Close-DSSession] No session was previously established.
Updating Service Credential
PS C:\Users\testaccount1>
```

Running a script integration of the Devolutions Secret Management module