

Integration Pack for Secure PowerShell

For Microsoft System Center Orchestrator

For System Center 2022 and later, you must use the 64-bit version of the integration pack, which has the name **Kelverion_IP_Secure_PowerShell_x64_1.0**

User Guide

Version 1.0

Kelverion Integration Pack for Secure PowerShell

Copyright 2016 Kelverion Inc. All rights reserved.

Published: October 2026

Feedback

Send suggestions and comments about this document to support@kelverion.com

Contents

Kelverion Integration Pack for Secure PowerShell 4
System Requirements 4
Azure Key Vault Credential Provider Requirements 4
Registering and Deploying the Integration Pack 4
Licensing the Integration Pack 6
Integration Pack Configuration Options 6
Integration Pack JSON Configuration 8
Configuration Schema 8
Parameter 9
Output 10
Examples 12
Credential Provider Script 13
Azure Key Vault Support 14
Username and Password Support 14
WinRM Configuration 14
Remote PowerShell Sessions 15
Correlation ID 15

Integration Pack Activities 17
Common Configuration Instructions for All Activities 17
Activity Properties 17
General Tab 17
Properties Tab 17
Run Behavior Tab 18
Published Data 19

Create Session 20

Get Session 22

Remove Session 23

Run Script 24

Run Session Script 26

Kelverion Integration Pack for Secure PowerShell

The **Integration Pack for Secure PowerShell** is an add-on for System Center Orchestrator that can be used to run PowerShell scripts on remote machines, without storing credentials in the Orchestrator database or publishing credentials on the Orchestrator data bus.

System Requirements

The Integration Pack for Secure PowerShell requires the following software to be installed and configured prior to implementing the integration. For more information about installing and configuring Orchestrator, refer to the respective product documentation.

- Microsoft System Center Orchestrator 2022, 2025
- Microsoft .NET Framework 4.7.2
- Windows Remote Management (WinRM)
- Windows PowerShell 5.1 enabled for remoting (Enable-PSRemoting)

Azure Key Vault Credential Provider Requirements

The integration pack installs a credential provider PowerShell script that retrieves target machine credentials from your Azure Key Vault. This requires the following software to be installed and configured:

- Azure Subscription with access to Azure Key Vault
- Az.Accounts PowerShell module version 5.5.3
- Az.KeyVault PowerShell module version 6.5.0
- Orchestrator Runbook Server must be Azure Arc-enabled and must be configured with a system-assigned managed identity with Azure Key Vault access.

Registering and Deploying the Integration Pack

After you download the integration pack, you register the integration pack file with the Orchestrator management server and then deploy it to Runbook Servers and computers that have the Runbook Designer installed.

To register the integration pack:

1. On the management server, copy the **.OIP** file for the integration pack to a local hard drive or network share.
2. Confirm that the file is not set to **Read Only** to prevent unregistering the integration pack at a later date.
3. Start the **Deployment Manager**.
4. In the navigation pane of the Deployment Manager, expand **Orchestrator Management Server**, right-click **Integration Packs** to select **Register IP with the Orchestrator Management Server**. The **Integration Pack Registration Wizard** opens.

5. Click **Next**.
6. In the **Select Integration Packs or Hotfixes** dialog box, click **Add**.
7. Locate the **.OIP** file that you copied locally from step 1, click **Open** and then click **Next**.
8. In the **Completing the Integration Pack Wizard** dialog box, click **Finish**.
9. On the **End User Agreement** dialog box, read the Kelverion License Terms**,** and then click **Accept**.
10. The **Log Entries** pane displays a confirmation message when the integration pack is successfully registered.

To deploy the integration pack:

1. In the navigation pane of the **Deployment Manager**, right-click **Integration Packs**, click **Deploy IP to Runbook Server** or **Runbook Designer**.
2. Select the integration pack that you want to deploy, and then click **Next**.
3. Enter the name of the runbook server or computers with the Runbook Designer installed, on which you want to deploy the integration pack, click **Add**, and then click **Next**.
4. Continue to add additional runbook servers and computers running the Runbook Designer, on which you want to deploy the integration pack. Click **Next**.
5. In the **Installation Options** dialog box, configure the following settings.
6. To choose a time to deploy the integration pack, select the **Schedule installation** check box, and then select the time and date from the **Perform installation** list.
7. Click one of the following:
    a. **Stop all running runbooks before installing the integration pack** to stop all running runbooks before deploying the integration pack.
    b. **Install the Integration Packs without stopping the running Runbooks** to install the integration pack without stopping any running runbooks.
8. Click **Next**.
9. In the **Completing Integration Pack Deployment Wizard** dialog box, Click **Finish**.
10. When the integration pack is deployed, the **Log Entries** pane displays a confirmation message.

Licensing the Integration Pack

After you register and deploy the integration pack you must provide a valid Kelverion license before running any runbooks that contain activities from the integration pack.

To deploy the integration pack license file to System Center Orchestrator 2022 or later:

1. Copy the .KAL license file to %PROGRAMFILES%\Kelverion Automation\Licenses
2. Repeat for each Orchestrator Runbook Server and Runbook Designer host system.

Integration Pack Configuration Options

An Orchestrator configuration establishes the connection properties the integration pack uses to connect to target computers over PowerShell remoting. You can create more than one set of configuration options – for example one per environment – and select which one an activity during activity design time.

To set up a Secure PowerShell configuration:

1. In the Client, click the **Options** menu, and select **KA Secure PowerShell**. The **KA Secure PowerShell** dialog box appears.
2. On the **Configurations** tab, click **Add** to begin the configuration setup. The **Add Configuration** dialog box appears.
3. In the **Name** box, enter a name for the configuration.
4. Click the ellipsis button (…) next to the **Type** box and select **Secure PowerShell Configuration**.
5. In the **Configuration File Path** box, enter the path to the integration pack JSON configuration file containing credential provider and script configuration definitions. For details, please refer to Integration Pack Configuration Options. Default configuration file is *Kelverion.IntegrationPack.Secure.PowerShell.Configuration.json*, in the IP installation folder.
6. In the **Credential Script Path** box, enter the path to the credential provider PowerShell script. For details, please refer to Credential Provider Script. Default credential provider script is *Get-Credential-AzureKeyVault.ps1*, in the IP installation folder.
7. (Optional) In the **Application Name** box, enter the WS-Management application name segment of the connection URL. Default value is **WSMAN**.
8. (Optional) In the **Authentication** box, select the authentication mechanism used for the connection. Available values are **Basic**, **CredSSP**, **Default**, **Digest**, **Kerberos**, **Negotiate** and **NegotiateWithImplicitCredential**. Default value is **Default**.
9. (Optional) In the **Session Configuration Name** box, select the PowerShell session configuration name on the target computer to connect to. Default value is **Microsoft.PowerShell**.

10. (Optional) In the **Port** box, enter the WinRM port to connect to. Default value is **5986,** which is the typical port for an HTTPS WinRM listener.
11. (Optional) In the **Use SSL** box, specify whether the connection uses **HTTPS**. Default value is **True**.
12. (Optional) In the **Skip CA Check** box, specify whether to skip certificate authority validation for the target computer's certificate. Default value is **False**.
13. (Optional) In the **Skip CN Check** box, specify whether to skip certificate common name validation for the target computer's certificate. Default value is **False**.
14. (Optional) In the **Skip Revocation Check** box, specify whether to skip certificate revocation check for the target computer's certificate. Default value is **False**.
15. (Optional) In the **Open Timeout (ms)** box, enter the number of milliseconds to wait when opening the connection before timing out. Default value is **60000**.
16. (Optional) In the **Idle Timeout (ms)** box, enter the number of milliseconds a session may remain idle before the target computer closes it. Default value is **60000**.
17. Click **OK** to close the configuration dialog box, and then click **Finish**.

Integration Pack JSON Configuration

When running PowerShell scripts, it can be beneficial to provide script input parameters from the Orchestrator data bus. Similarly, you may wish to make script data available to the Orchestrator data bus, as published data. You can accomplish this by using the integration pack JSON configuration file:

- Maps integration pack activity inputs to PowerShell script parameters
- Maps PowerShell script data to activity published data outputs

The integration pack installs the *Kelverion.IntegrationPack.Secure.PowerShell.Configuration.json* configuration file in the same location where the integration pack binaries are installed.

There are two distinct sections in the configuration file:

- **CredentialProviderConfig** section – defines parameters and outputs for Credential Provider Scripts. Script parameters/outputs will appear as input properties/published data in the **Create Session** and **Run Script** activities.
- **ScriptConfig** section – defines parameters and outputs for PowerShell scripts that you run with the **Run Script** and **Run Session Script** activities. Script parameters/outputs will appear as input properties/published data in these activities.

The installed configuration file contains sample script configurations. You can create additional configurations or edit the existing ones. Make sure to back up the configuration file before making changes.

Important: We strongly recommend users locate their custom JSON configuration file(s) in a location outside of the IP folder, such as a network share, that is visible from all Runbook Server and Runbook Designer machines. The installed JSON configuration file in the IP folder will be replaced each time you perform an IP upgrade, so we recommend you change the file name and location.

Configuration Schema

The **CredentialProviderConfig** and **ScriptConfig** are similar elements that define configuration schema for credential provider scripts, and scripts used in the Run [Session] Script activities, respectively.

CredentialProviderConfig

The **CredentialProviderConfig** element contains configurations that define inputs and outputs for Credential Provider Scripts:

```
"CredentialProviderConfig": [
{
    "Name": "Credential Provider",
    "Parameters": [...],
    "Outputs": [...]
},
...]
```

Important: At least one credential provider configuration must be defined, corresponding to the credential provider PowerShell script specified in the **Credential File Path**, in the Integration Pack Configuration Options.

Important: Credential provider configurations must always define a PSCredential output, returned by the credential provider script. The IP will use this to authenticate when opening remote sessions on target machines. ***Make sure to never publish the PSCredential output. IsPublished should always be set to "false" for this output. Otherwise, credential information may be published on the Orchestrator data bus.***

ScriptConfig

The **ScriptConfig** element contains script configurations for script running activities:

```
"ScriptConfig": [
{
    "Name": "Write Output Sample",
     "Parameters": [...],
     "Outputs": [...]
},
...]
```

You can specify the following properties in the **CredentialProviderConfig** and **ScriptConfig** elements:

- **Name** – specifies a unique name for this configuration.
- **Parameters** – Defines zero or more **Parameter** elements. Activities will provide design time input properties for each defined parameter. Activity input runtime values will be available in the PowerShell script as PowerShell parameters.
- **Outputs** – Defines zero or more **Output** elements. At runtime, activities will publish the PowerShell script outputs on the Orchestrator data bus.
- **ScriptPath** – Specifies a PowerShell script corresponding to this configuration.
    - For **CredentialProviderConfig -** Specifies the credential provider PowerShell script corresponding to this configuration. This can be an absolute path or a unique file name. The **ScriptPath** file name in at least one of the credential provider configurations must match the specified **Credential File Path** file name, in the Integration Pack Configuration Options.
    - For **ScriptConfig -** Specifies the location of a PowerShell script to be used instead of specifying the **Script** property in the **Run Script** and **Run Session Script** activities. This can be an absolute path, or a unique file name for a script located in the IP installation folder. When specified, the **Script** input property becomes optional, and it can be used to override the script specified by ScriptPath.

Parameter

Defines a script parameter. For each defined script parameter, the activity will provide an activity input property.

```
"Parameters": [
```

```
{
    "Name": "RequiredInput",
    "InputName": "Required Input",
    "DataType": "System.String"
    ...
},
...]
```

You can specify the following parameter properties:

- **Name** – specifies the script parameter name. For example, when specifying the name **RequiredInput** in the JSON configuration file, the PowerShell $RequiredInput parameter will be available to be used by your script at runtime.
- **InputName** – specifies the activity input name, corresponding to this parameter.
- **DataType** – specifies the parameter data type. Supported values are:
    - System.String
    - System.Security.SecureString
    - System.Boolean
    - System.DateTime
    - System.Int32
    - System.Int64
    - System.Single
    - System.Double
    - System.Decimal
- **IsPassword** – specifies whether the string input value will be obfuscated in the Orchestrator Runbook Designer. Supported values are **true** and **false**.
- **IsRequired** – specifies whether the activity input property is required. Supported values are **true** and **false**.
- **Values** – optionally defines a fixed list of values for the activity input property, shown as a drop-down list in the Runbook Designer instead of a free-text box.

Important: When defining parameters, avoid input names which have already been defined in the activity or activity configuration options, as this can result in configuration options values being overwritten by activity values at runtime.

Output

Defines a PowerShell script output. The integration pack will expect to find a PowerShell property with this name in the list of PowerShell objects returned by the script. Output values will be published at runtime on the Orchestrator data bus.

```
"Outputs": [
{
    "Name": "OutputFromRequiredInput",
    "PublishName": "Output From Required Input",
    "IsPublished": true,
    "DataType": "System.String"
},{
    "Name": "OutputFromOptionalInput",
```

```
    "PublishName": "Output From Optional Input",
    "IsPublished": true,
    "DataType": "System.Int32"
},
... ]
```

You can specify the following output properties:

- **Name** – specifies the output name. This name must match the PowerShell output name.
- **PublishName** - specifies the Orchestrator published data name. The integration pack will publish an activity output with this name on the Orchestrator data bus.
- **DataType** – specifies the output data type. Supported values are:
    - System.String
    - System.Boolean
    - System.DateTime
    - System.Int32
    - System.Int64
    - System.Double
    - System.Decimal
- **IsPublished** – specifies whether this output is published on the Orchestrator data bus. Supported values are **true** and **false**.
  Important: Never set **IsPublished** to **true** for the **PsCredential** output of the credential provider script, to avoid publishing sensitive information.

Examples

The integration pack is installed with two sample script configurations:

- Write Output Sample
- Get Process Sample

Write Output Sample

This sample configuration defines the following inputs:

- **Required Input** – available as a required input property in Run [Session] Script activities. You can access this value in the PowerShell script by using the **$RequiredInput** parameter.
- **Optional Input** – available as an optional input property in Run [Session] Script activities. You can access this value in the PowerShell script by using the **$OptionalInput** parameter.

This sample configuration defines the following outputs:

- **Output From Required Input** – published by Run [Session] Script activities. In the PowerShell script, you must assign data to the **OutputFromRequiredInput** property of the returned object.
- **Output From Optional Input** – published by Run [Session] Script activities. In the PowerShell script, you must assign data to the **OutputFromOptionalInput** property of the returned object.

A simple PowerShell script that echoes the inputs as outputs:

```
return [PSCustomObject] @{
    OutputFromRequiredInput = $RequiredInput
    OutputFromOptionalInput = $OptionalInput
}
```

The input values are accessed by using the `$RequiredInput` and `$OptionalInput` properties, respectively.

The script returns a single PSCustomObject. Properties of this object are published as individual properties on the Orchestrator data bus. To publish activity data from the script, you must assign values to the `OutputFromRequiredInput` and `OutputFromOptionalInput` properties of the returned PSCustomObject.

Get Process Sample

This sample configuration does not define any inputs.

This sample configuration defines the following outputs:

- **Process Name** – published by Run [Session] Script activities. Maps to the **ProcessName** property in the PS object returned by the script.

- **Process Id** – published by Run [Session] Script activities. Maps to the **Id** property in the PS object returned by the script.
- **Handle Count** – published by Run [Session] Script activities. Maps to the **Handles** property in the PS object returned by the script.
- **Working Set** – published by Run [Session] Script activities. Maps to the **WorkingSet64** property in the PS object returned by the script

The following script returns the first 30 processes running on the target machine:

```
Get-Process | select -First 30
```

The script returns a collection of System.Diagnostics.Process objects, published as multiple records on the Orchestrator data bus. For each Process object, only the properties defined as outputs in the script configuration are published.

Credential Provider Script

The **Create Session** and **Run Script** activities use a credential provider PowerShell script to retrieve the credentials necessary to connect to the target machine. The **Credential Script Path** configuration property specifies the location of the credential provider script.

The script runs locally on the Orchestrator Runbook Server, not on the target computer, and must return credentials for remote machines. Script running activities will use the provided credentials to authenticate when creating remote sessions.

The credential provider script parameters and outputs **must** match the parameter and output names for the corresponding **CredentialProviderConfig**, in the Integration Pack JSON Configuration File. Specified parameters become activity input properties in the **Create Session** and **Run Script** activities.

Important: The script must return exactly one PowerShell object with a PsCredential property of type System.Management.Automation.PSCredential:

```
$result = [PSCustomObject] @{ PsCredential = <PSCredential object> }
return $result
```

You can replace the installed credential provider script with your own script instead, as long as it follows the same contract: declare its parameters and output in CredentialProviderConfig and return a PSCredential. Make sure to back up the configuration file before making changes.

Important: We strongly recommend users locate their custom credential provider script(s) in a location outside of the IP folder, such as a network share, that is visible from all Runbook Server and Runbook Designer machines. The installed credential provider scripts in the IP folder will be replaced each time you perform an IP upgrade, so we recommend you change the file name and location.

Azure Key Vault Support

The integration pack installs the *Get-Credential-AzureKeyVault.ps1* script file in the same location where the integration pack binaries are installed. This script retrieves credentials from Azure Key Vault, which then are used by integration pack activities when opening remote PowerShell sessions.

When using this credential provider script, you will have to specify the following properties in the **Create Session** and **Run Script** activities:

- **Azure Key Vault Name** – Azure Key Vault containing credentials
- **Username Secret Name** – Name of the Username secret in the Key Vault
- **Password Secret Name** – Name of the Password secret in the Key Vault

Important: The Orchestrator runbook server must be configured as an **Azure Arc** machine, with a system-assigned managed identity. The managed identity must be granted **Get** permissions on the Key Vault to retrieve vault secrets. This allows the credential provider script to connect and retrieve credentials from Azure Key Vault.

Important: The Orchestrator runbook service account must be a member of the local security group **Hybrid agent extension applications**, in order to request Azure managed identity tokens. This group is created when the runbook server is configured as an Azure Arc machine. Make sure to restart the Orchestrator runbook service after making the group membership change.

Important: When running the Orchestrator test console, the logged-in user account must be a member of the local security group **Hybrid agent extension applications**, in order to request Azure managed identity tokens. This group is created when the runbook server is configured as an Azure Arc machine. Make sure to re-log after making the group membership change.

Username and Password Support

The integration pack installs the *Get-Credential-UsernamePassword.ps1* script file in the same location where the integration pack binaries are installed. This credential provider script can be used for testing or for cases when Azure Key Vault is not available. The script accepts username and password parameters and returns the corresponding PsCredential object, which then is used by integration pack activities when opening remote PowerShell sessions.

When using this credential provider script, you will have to specify the following properties in the **Create Session** and **Run Script** activities:

- **Username** – username for remote session authentication
- **Password** – password for remote session authentication

WinRM Configuration

The integration pack uses Windows Remote Management to run PowerShell scripts remotely. By default, Windows server machines may have minimal WinRM configuration, with HTTP listeners configured to run on port 5985. To use HTTPS, you may have to configure a new WinRM listener for HTTPS.

Requirements for running remote PowerShell with WinRM, for each remote target machine:

- Windows PowerShell 5.1
- WinRM remoting enabled (Enable-PSRemoting)

Requirements for running WinRM over HTTPS, for each remote target machine:

- Create an HTTPS listener. The HTTPS listener must be configured with a certificate issued by an approved certificate authority. When using self-signed certificates for testing, you must configure the certificate to be trusted on the Orchestrator server or, you can use the Skip CA/CN Chek properties in the IP configuration options.
- Configure firewall. Typical HTTPS listener port is 5986.
- Grant remote access – the connecting account must be a member of the **Remote Management Users** local group, or a local administrator, on each remote machine. This applies to:
    - The Orchestrator runbook service account. Restart the service after making the group membership changes.
    - The logged-on account, for running the Orchestrator runbook tester. Re-log after making the group membership changes.

Remote PowerShell Sessions

Create Session, Get Session, Run Session Script and Remove Session activities share a common concept, a **session** which represents a remote PowerShell session on a remote machine, that stays open across multiple activity executions until it is explicitly closed.

The **Create Session** activity creates the session and publishes a new **Session ID**. You pass the Session ID to the **Run Session Script** activity to run one or more scripts in the already-open session, without reconnecting or re-authenticating. The **Get Session** activity is used to retrieve existing session information. And finally, the **Remove Session** activity is used to close the session.

Sessions are held in memory by the runbook process that created them and do not persist in the Orchestrator database. The only session information published on the Orchestrator data bus is the **Session ID** and the **Correlation ID** (see below).

Orchestrator runbook activities run in the same runbook process, so a session created by the **Create Session** activity will be available to following activities in that same process. A session does not survive a Runbook Server restart and remains open on the target computer until it is explicitly closed with the **Remove Session** activity, or the connection Idle Timeout elapses. Always pair Create Session activities with matching Remove Session, including on error paths, to avoid leaking open connections.

Correlation ID

A **Correlation ID** is a free-form string identifier that you provide in activities that open remote PowerShell sessions – **Create Session** and **Run Script** activities. It exists so you can map activity execution on the Orchestrator side to script execution on the remote machine. The Correlation ID

value is available in the remote session as $PSSenderInfo.ApplicationArguments.CorrelationId, which is useful for matching remote-side executions with the originating runbook job.

Integration Pack Activities

This integration pack adds the KA Secure PowerShell category to the **Activities** pane in the Orchestrator Runbook Designer. This category contains the following activities:

- Create Session
- Get Session
- Remove Session
- Run Script
- Run Session Script

Common Configuration Instructions for All Activities

The following configuration instructions apply to all activities in this integration pack. Links to this section are included in the configuration instructions for each activity.

Activity Properties

Each activity has a set of required or optional properties that define the configuration of that activity. This includes how it connects to other activity or how the activity performs its actions. You can view or modify activity properties in the Orchestrator Client.

To configure the properties for an activity:

1. Double-click the activity. Alternatively, you can right-click the activity, and then click **Properties**.
2. To save your configuration entries, click **Finish**.

In the activity properties dialog box, several tabs along the left side provide access to general and specific settings for the activity. Although the number of available tabs for activity properties differs from activity to activity, all activities will have a **General** tab, a **Properties** tab and/or **Filters** tab, and a **Run Behavior** tab. Some activities may have additional tabs.

General Tab

This tab contains the **Name** and **Description** properties for the activity. By default, the **Name** of the activity is the same as its activity type, and the **Description** is blank. You can modify these properties to create more descriptive names or provide detailed descriptions of the actions of the activity.

Properties Tab

These tabs contain properties that are specific to the activity.

All activities in this integration pack have the **Configuration Name** property at the top of the **Properties** tab. This property is used to specify the connection to a Secure PowerShell server.

To configure the Configuration Name property:

1. Click the ellipsis **(…)** button next to the **Name** field, and then select the applicable connection name. Connections displayed in the list have been previously configured as described in Configuring the Integration Pack.

Run Behavior Tab

This tab contains the properties that determine how the activity handles multi-value published data and what notifications will be sent if the activity fails or runs for an excessive period of time.

Multi-Value Published Data Behavior

The Get activities retrieve information from another activity or outside source and can return one or more values in the published data. For example, when you use the Get Collection Member activity, the data output from that activity might be a list of computers that belong to the specified collection.

By default, the data from the Get activity will be passed on as multiple individual outputs. This invokes the next activity as many times as there are items in the output. Alternatively, you can provide a single output for the activity by enabling the **Flatten** option. When you enable this option, you also choose a formatting option:

- **Separate with line breaks**. Each item is on a new line. This format is useful for creating human-readable text files for the output.
- **Separate with _** . Each item is separated by one or more characters of your choice.
- **Use CSV format**. All items are in CSV (comma-separated value) format. This format is useful for importing data into spreadsheets or other applications.

The activity will produce a new set of data every time it runs. The **Flatten** feature does not flatten data across multiple instances of the same activity.

Event Notifications

Some activities are expected to take a limited amount of time to complete. If they do not complete within that time they may be stalled or there may be another issue preventing them from completing. You can define the number of seconds to wait for completion of the action. After this period a platform event will be sent, and the issue will be reported. You can also choose whether to generate a platform event if the activity returns a failure.

To be notified when the activity takes longer than a specified time to run or fails to run:

1. In the **Event Notifications** box, enter the **number of seconds** of run time before a notification is generated.
2. Select **Report if activity fails to run** to generate run failure notifications.

For more information about Orchestrator events, see the “Event Notifications " topics in the Runbook Properties (https://technet.microsoft.com/en-us/library/hh489610.aspx#Event Notifications).

Published Data

Published data is the foundation of a working runbook. It is the data produced as a result of the actions of an activity. This data is published to an internal data bus that is unique for each runbook. Subsequent activities in the runbook can subscribe to this data and use it in their configuration. Link conditions also use this information to add decision-making capabilities to runbooks.

An activity can only subscribe to data from the activities that are linked before it in the runbook. You can use published data to automatically populate the property values needed by activities.

To use published data:

1. Right-click the property value box, click **Subscribe**, and then click **Published Data**.
2. Click the **Activity** drop-down box and select the activity from which you want to obtain the data.
3. To view additional data elements common to all runbooks, select **Show Common Published Data**.
4. Click the published data element that you want to use, and then click **OK**.

For a list of the data elements published by each activity, see the Published Data tables in the activity topic.

Create Session

The **Create Session** activity can be used in a runbook to open a remote PowerShell session to the specified target computer. The session will remain open for later use by the **Run Session Script** and **Get Session** activities. When all session work is complete, use the **Remove Session** activity to close and clean up the session.

Required Properties

You must configure the following properties:

Computer Name	FQDN, hostname or IP address of the target computer to connect to.
Correlation ID	Identifier used to correlate the session with target environment. Available to scripts running in this session as: $PSSenderInfo.ApplicationArguments.CorrelationId.

Required Azure Key Vault Credential Provider Properties

When using the installed Azure Key Vault credential provider script, you must configure the following properties to retrieve credentials from Azure Key Vault. If your IP configuration options point to a different **Credential Script Path**, you may see different credential provider properties.

Azure Key Vault Name	Name of the Azure Key Vault holding the target credential.
Username Secret Name	Name of the Key Vault secret holding the username.
Password Secret Name	Name of the Key Vault secret holding the password.

Required Username and Password Credential Provider Properties

When using the installed username and password credential provider script, you must configure the following properties. If your IP configuration options point to a different **Credential Script Path**, you may see different credential provider properties.

Username	Username for remote session authentication.
Password	Password for remote session authentication.

Published Data

The activity publishes the following activity-specific data items.

Session ID	Unique identifier of the created session.
Computer Name	The target computer the session was opened on.

Correlation ID	The Correlation ID for the session.
Session Created At (UTC)	Date and time (UTC) the session was created.

Get Session

The **Get Session** activity can be used in a runbook to retrieve an existing session by its Session ID.

Note: Sessions are held in memory by the runbook process that created them and are not persisted to the Orchestrator database. If the session has already been removed, or the Runbook Server that created it was restarted, the activity fails with a session-not-found error.

Required Properties

You must configure the following property:

Session ID	Identifies the session to retrieve. This must match the Session ID published by a previous Create Session activity.

Published Data

The activity publishes the following activity-specific data items.

Session ID	Unique identifier of the session.
Computer Name	The target computer the session was opened on.
Correlation ID	The Correlation ID for the session.
Session Created At (UTC)	Date and time (UTC) the session was created.

Remove Session

The **Remove Session** activity can be used in a runbook to close an existing session and release the underlying PowerShell remoting connection to the target computer.

Note: If the session was previously removed and Session ID does not match an open session, the activity logs a warning and completes successfully.

Required Properties

You must configure the following property:

Session ID	Identifies the session to remove. Must match the Session ID published by a previous Create Session activity.

Published Data

This activity does not publish any activity-specific data items.

Run Script

The **Run Script** activity can be used in a runbook to open a remote PowerShell connection to a target computer, run a script, and then close the connection, all in a single activity. The remote session is closed when the activity is completed, and there is no session information stored in the runbook process memory.

Note: Always return small objects from your scripts, containing only the output data that is necessary for runbook operation. Returning large PowerShell objects can bloat the Orchestrator data bus with unnecessary, or unintended, published data or, it may result in serialization errors. For example, instead of returning an entire $PSSenderInfo object, and risk publishing sensitive information, you can instead only return the individual string properties that are needed by the runbook:

```
return [PSCustomObject] @{
    ConnectedUser = $PSSenderInfo.ConnectedUser
    RunAsUser = $PSSenderInfo.RunAsUser
    ConnectionString = $PSSenderInfo.ConnectionString
}
```

Required Properties

You must configure the following properties:

Computer Name	FQDN, hostname or IP address of the target computer to connect to.
Correlation ID	Identifier used to correlate the session with target environment. Available to scripts running in this session as: $PSSenderInfo.ApplicationArguments.CorrelationId.
Script Configuration	Specifies a script configuration from the JSON configuration file. Based on this configuration, the activity provides additional input properties and published data. Select <NONE> to run a script with no custom parameters or outputs. For details, refer to Integration Pack JSON Configuration.
Script	The PowerShell script to run. If you are using the JSON configuration file to define custom script parameters, activity input runtime values will be available in the script as PowerShell parameters. If you are using the JSON configuration file to define custom script outputs, output script values will be published on the Orchestrator data bus. If the selected script configuration defines a **ScriptPath** in the JSON configuration file, this property becomes optional. When not specified, the activity will run the script located at **ScriptPath**. Specifying a script in this case will override the script located at ScriptPath.

	If the selected script configuration does not define a ScriptPath, this property is required.

Required Azure Key Vault Credential Provider Properties

When using the installed Azure Key Vault credential provider script, you must configure the following properties to retrieve credentials from Azure Key Vault. If your IP configuration options point to a different **Credential Script Path**, you may see different credential provider properties.

Azure Key Vault Name	Name of the Azure Key Vault holding the target credential.
Username Secret Name	Name of the Key Vault secret holding the username.
Password Secret Name	Name of the Key Vault secret holding the password.

Required Username and Password Credential Provider Properties

When using the installed username and password credential provider script, you must configure the following properties. If your IP configuration options point to a different **Credential Script Path**, you may see different credential provider properties.

Username	Username for remote session authentication.
Password	Password for remote session authentication.

Published Data

The activity publishes the following activity-specific data items.

Note: If the selected Script Configuration defines custom Outputs, the activity will publish data according to those output definitions.

Correlation ID	The Correlation ID supplied to the activity.
Result Object Count	Number of PowerShell objects returned by the script.
Result Object	JSON representation of the returned PowerShell object. Published when no script configuration is specified, or if the specified script configuration does not define any outputs.
Result Text	Text representation of the returned PowerShell object. Published when no script configuration is specified, or if the specified script configuration does not define any outputs.

Run Session Script

The **Run Session Script** activity can be used in a runbook to run a script in a session previously opened with Create Session, without opening a new connection.

Note: Always return small objects from your scripts, containing only the output data that is necessary for runbook operation. Returning large PowerShell objects can bloat the Orchestrator data bus with unnecessary, or unintended, published data or, it may result in serialization errors. For example, instead of returning an entire $PSSenderInfo object, and risk publishing sensitive information, you can instead only return the individual string properties that are needed by the runbook:

```
    return [PSCustomObject] @{
        ConnectedUser = $PSSenderInfo.ConnectedUser
        RunAsUser = $PSSenderInfo.RunAsUser
        ConnectionString = $PSSenderInfo.ConnectionString
}
```

Required Properties

You must configure the following properties:

Session ID	Identifies the previously created session. Must match the Session ID published by a previous **Create Session** activity.
Script Configuration	Specifies a script configuration from the JSON configuration file. Based on this configuration, the activity provides additional input properties and published data. Select <NONE> to run a script with no custom parameters or outputs. For details, refer to Integration Pack JSON Configuration.
Script	The PowerShell script to run. If you are using the JSON configuration file to define custom script parameters, activity input runtime values will be available in the script as PowerShell parameters. If you are using the JSON configuration file to define custom script outputs, output script values will be published on the Orchestrator data bus. If the selected script configuration defines a **ScriptPath** in the JSON configuration file, this property becomes optional. When not specified, the activity will run the script located at **ScriptPath**. Specifying a script in this case will override the script located at ScriptPath. If the selected script configuration does not define a ScriptPath, this property is required.

Published Data

The activity publishes the following activity-specific data items.

Note: If the selected Script Configuration defines custom Outputs, the activity will publish data according to those output definitions.

Correlation ID	The session Correlation ID.
Result Object Count	Number of PowerShell objects returned by the script.
Result Object	JSON representation of the returned PowerShell object. Published when no script configuration is specified, or if the specified script configuration does not define any outputs.
Result Text	Text representation of the returned PowerShell object. Published when no script configuration is specified, or if the specified script configuration does not define any outputs.