INTEGRATION MODULE FOR MICROSOFT SQL SERVER

For Kelverion Runbook Studio and Azure Automation

User Guide

Version 2.4

Kelverion Integration Module for SQL Server

Released: September 2023

The Kelverion Integration Module for SQL Server is Microsoft Azure Certified

Feedback

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

Contents

Getting Started

The following sections outline how to deploy and configure the Kelverion Integration Module for SQL Server.

System Requirements

The Integration Module for SQL Server requires the following software to be installed and configured prior to implementing the integration. For more information about installing and configuring Microsoft SQL Server, refer to the respective product documentation.

- Kelverion Runbook Studio 5.6
- Microsoft .NET Framework 4.7.2

One of these database servers:

- Microsoft SQL Server 2017, 2019, 2022
- Azure SQL Server

Deploying the Integration Module

The easiest way to install and deploy the Integration Module for Oracle is from the PowerShell Gallery, but you can also download the module from Kelverion and perform the steps manually.

You must install and deploy the Integration Module to each Azure Automation Account and Hybrid Worker host system that you plan to use to run your runbooks. You must also install the Integration Module on any Runbook Studio host systems that you will be using to build and manage your runbooks.

Using the PowerShell Gallery

Using the commands in the **PowerShellGet** module you can download the Kelverion Integration Module for Microsoft SQL Server from the PowerShell Gallery and install it on your local computer. You can also deploy the module directly from the PowerShell Gallery to any of your Azure Automation Accounts.

Install the Integration Module on your local computer:

1. Confirm that the PowerShellGet module is installed.
2. Start a PowerShell window as Administrator and run the command:

   ```
   Install-Module -Name Kelverion.SqlServer -Scope AllUsers
   ```

Upload the Integration Module to an Azure Automation Account:

1. Go to the PowerShell Gallery.
2. Click the **Azure Automation** tab.
3. Click **Deploy to Azure Automation**. You will be directed to Microsoft Azure.
4. Select the **Automation Account** that you want to deploy the module to.
5. Click **OK**.

Manual Installation

Alternatively, you can download the Integration Module package from Kelverion and deploy it manually to your local computer, hybrid workers and Automation Accounts.

The download package from Kelverion includes a **.zip** file containing the Integration Module as well as the User Guide and Release Notes. The following instructions assume that you have unzipped the download package and have access to the **.zip** file containing the Integration Module.

Install the Integration Module on your local computer:

1. Copy the **Kelverion.SqlServer.zip** file to your local computer.
2. Right-click on the file and select **Properties**.
3. Click the **General** tab. If necessary, click **Unblock**, and then click **OK**.
4. Unzip the **Kelverion.SqlServer.zip** file.
5. Copy the **Kelverion.SqlServer** folder to a location in the `%PsModulePath%` path.

Important: When installing the Integration Module on a Hybrid Worker, you must use a location that is accessible to all users of the computer.

Upload the Integration Module to an Azure Automation Account:

1. Sign into Microsoft Azure.
2. Open the Automation Account that you want to upload the module to.
3. Click **Modules** under Shared Resources. The list of installed modules is displayed.
4. Click **Add a module** at the top of the list.
5. In the **Upload File** box, select the **Kelverion.SqlServer.zip** file that you downloaded.
6. Click **OK**. Importing the module may take several minutes.

Licensing the Integration Module

Licenses for Kelverion Integration Modules are managed and deployed using *Kelverion Runbook Studio* and *Automation Connection Assets*.

Register an Integration Module license with Runbook Studio:

1. Open **Kelverion Runbook Studio**.
2. In the **File** tab, click **About**.
3. Click **License Information**.
4. Click the **Integration Modules** tab, and then click **Add License**.
5. Select the integration module license file (*.kaml) and click **Open**.
6. You should see your entitlements displayed in the list.
7. Click **OK**.

Important: Entitlements will not display until after the Integration Module has been installed on the Runbook Studio computer.

Create a Connection Asset with a license key and upload it to Azure:

1. On the **Home** tab, click **Sign In**. The Sign in dialog appears.
2. Sign into your account.
3. In the **Active Azure Automation Account** box, select that account that you want to add the connection asset. To.
4. Click **New Asset** and then click **Connection**. The New Connection dialog appears,
5. In the **Name** field, enter a name to identify the connection.
6. In the **Connection Type** field, select the desired connection type.
7. Enter the appropriate connection information in the provided fields.
8. Click **OK**.

Update all Connection Assets license keys and upload them to Azure:

1. On the **Home** tab, click **Sign In**. The Sign in dialog appears.
2. Sign into your account.
3. In the Explorer panel, click the **Azure (Online)** group.
4. Right-click the Azure Automation Account that contains the connection assets you want to update, and then and then click **Update License Keys**. A summary is displayed.

Working With SQL Server Always Encrypted Certificates

Column Master Keys are key-protecting keys used in SQL Server Always Encrypted to encrypt column encryption keys. Column master keys are stored in trusted key stores, and the keys need to be accessible to applications that need to encrypt or decrypt data, including Runbook Studio, Azure Automation and Hybrid Workers.

The following tasks must be completed if you are going to work with encrypted database data in your runbooks:

1. Export Always Encrypted certificates from your SQL Server host machines.
2. Importing Always Encrypted Certificates for Azure Automation Account Access.
3. Importing Always Encrypted Certificates for Hybrid Worker Access.

Important: This version of the Integration Module for SQL Server only supports certificates stored in Local Key Stores, such as the **Windows Certificate Store**. It does not support certificates stored in Azure Key Vault.

Exporting Always Encrypted Certificates from SQL Server

Encrypted column master keys may be stored in the *local machine* certificate store or *current user* certificate store. In either case, you need to export the certificate with the private key and import it to all machines that host Hybrid Workers as well as to each one of your Automation Accounts that are expected to encrypt or decrypt data stored in encrypted columns.

For Always Encrypted certificates stored in the current user certificate store:

1. On the server where you have Always Encrypted enabled, select **Run** from the **Start** menu and then enter **certmgr.msc**. The Certificate Manager tool for the current user appears.
2. Find the Always Encrypted certificate (probably located in Current Users\Personal\Certificates).
3. Right click on the Always Encrypted certificate and select **All tasks** then **Export**. The Certificate Export Wizard appears.
4. In the Export Private Key page, select **Yes, export the private key**, and click **Next**.
5. In the Export File Format page, accept all the defaults and click **Next**.
6. In the Security page, select **Password**. Enter and confirm a password and click **Next**.
7. In the File to Export page, enter a file name and click **Next**.
8. Click **Finish**.

For Always Encrypted certificates stored in the local machine certificate store:

1. On the server where you have Always Encrypted enabled, select **Run** from the **Start** menu and then enter **certlm.msc**. The Certificate Manager tool for the current user appears.
2. Find the Always Encrypted certificate (probably located in Local Computer\Personal\Certificates).

3. Right click on the Always Encrypted certificate and select **All tasks** then **Export**. The Certificate Export Wizard appears.
4. In the Export Private Key page, select **Yes, export the private key**, and click **Next**.
5. In the Export File Format page, accept all the defaults and click **Next**.
6. In the Security page, select **Password**. Enter and confirm a password and click **Next**.
7. In the File to Export page, enter a file name and click **Next**.
8. Click **Finish**.

Importing Certificates for Azure Automation Account Access

Encrypted column master keys must be uploaded to each Azure Automation account that will be used to run runbooks that will encrypt and/or decrypt data stored in encrypted columns.

1. Open Runbook Studio.
2. On the Home ribbon, click **Sign In.** Enter your Microsoft credentials.
3. On the Home ribbon, select the Automation account that you want to upload the certificate to.
4. On the Home ribbon, click **New Asset** and then click **Certificate**. The New Certificate dialog appears.
5. In the **Name** box, enter a name for the certificate.
6. In the **Certificate File** box, click **…** and select the Always Encrypted certificate file that you exported from the SQL Server machine's certificate store.
7. In the **Password** box, enter the password used to protect the certificate.
8. Click **OK**.

Importing Certificates for Hybrid Worker Access

Encrypted column master keys must be imported into the *local machine* certificate store for each of your hybrid worker host machines that will be used to run runbooks that will encrypt and/or decrypt data stored in encrypted columns.

1. On the Runbook Studio machine, select **Run** from the **Start** menu and then enter **certlm.msc**. The Certificate Manager tool for the *local machine* appears.
2. Under **Certificates – Local Computer**, right click **Personal** and select **All Tasks** and then **Import**. The Certificate Import Wizard appears.
3. Click **Next**.
4. In the File to Import page, enter the file path of the Always Encrypted certificate that you exported from the SQL Server machine and click **Next**.
5. In the Private key protection page, enter the password used to protect the certificate. Accept default import options and click **Next**.
6. In the Certificate store page, accept all defaults and click **Next**.
7. Click **Finish**.

Working with Activities in Runbook Studio

The following sections outline some of the common configuration options that are available to you when working with the activities in the Kelverion Integration Module for Microsoft SQL Server.

The integration module includes the following activities:

Delete-SQLRow	Delete rows from a database table or view using filter criteria to determine which rows should be deleted
Insert-SQLRow	Insert a row into database table or view
Invoke-SQLProcedure	Invoke a stored procedure and return the results
Invoke-SQLCommand	Invoke an SQL statement and return the results as a sequence of comma separated values
Select-SQLRow	Retrieve rows from a database table or view and return the results as a sequence of PSObject instances
Update-SQLRow	Update rows in a database table or view using filter criteria to determine which rows should be updated.

The advanced discovery capabilities provided by the activities in this integration module are only supported when authoring runbooks in Kelverion Runbook Studio.

When you publish your runbooks from Kelverion Runbook Studio to Azure Automation or when you generate PowerShell code snippets for Service Management Automation, Runbook Studio will automatically convert the dynamically generated parameters and filters of Smart activities into the parameters provided by the underlying command activities.

Smart Connections

In Kelverion Runbook Studio you can configure one or more Smart Connections to establish reusable links between Runbook Studio and specific SQL Server instances. You can create as many Smart Connections as you require, specifying links to multiple databases. You can also create multiple Smart Connections to the same database to allow for differences in security privileges for different user accounts.

Add a Smart Connection in Kelverion Runbook Studio:

1. On the **Home** tab, click **Smart Connections**. The Smart Connections dialog appears.
2. Click **Add a connection** at the top of the list.
3. In the **Name** box, enter the name for the connection.
4. In the **Connection Type** box, select *Kelverion.SqlServer***.**
5. In the **ServerName** box, type the name of the SQL Server host that you want to connect to.

6. In the **UseWindowsAuthentication** box, select *True* or *False.*
7. If you are not using windows authentication, in the **UserName** and **Password** boxes, type the credentials to use to connect to the database server.
8. Click **OK**, and then click **OK** again.

Global Connection Assets

The recommended way to pass connection information to your activities in your runbooks is to use Global Connection Assets. Global connection assets let you securely define connection information in Azure which can then be retrieved on demand using either the *Get-AutomationConnection* cmdlet or Connection Asset Data Source.

Add a global connection asset in Runbook Studio:

1. On the **Home** tab, click **Sign In**. The Sign In dialog appears.
2. Sign into your account.
3. In the **Active Azure Automation Account** box, select the account that you want to add the connection asset to.
4. Click **New Asset** and then click **Connection**. The New Connection dialog appears.
5. In the **Name** box, enter a name for the configuration.
6. In the **Connection Type** box, select *Kelverion.SqlServer***.**
7. In the **ServerName** box, type the name of the SQL Server host that you want to connect to.
8. In the **UserName** and **Password** boxes, type the credentials to use to connect to the SQL Server database.
9. Click **OK**.

Activity Properties

All activities in the Kelverion Integration Module for Microsoft SQL Server have the following properties:

Property	Description
Label	A unique label that identifies the activity in the runbook. Runbook Studio will provide a default name for each activity, but you can provide your own labels to make their role in the runbook more obvious.
Description	An optional description of the activity. Providing a description is a great way to let everyone understand the function of the activity in the runbook.
Checkpoint	Indicates whether a checkpoint is set in the runbook workflow after the activity runs. Checkpoints are only available for Graphical PowerShell Workflow runbooks. If the runbook uses Azure cmdlets, you should follow best practices and follow a

	check-pointed activity with an *Add-AzureRMAccount* in case the runbook is suspended and restarts from this checkpoint on a different worker.

Smart Discovery

When designing runbooks in Kelverion Runbook Studio, you will notice that the activities in the Kelverion Integration Module for Microsoft SQL Server include a **Discovery** panel instead of the **Parameter Sets** panel that is present for standard command activities. This is because the activities in the Kelverion Integration Module for Microsoft SQL Server support interactive discovery of the database assets in your on-premises and cloud environments.

All activities in the Kelverion Integration Module for Microsoft SQL Server have a **Connection** option on the **Discovery** panel which lets you specify how Runbook Studio should connect to SQL Server.

When connected to SQL Server, Runbook Studio will provide additional discovery options, such as Database and Table Name, which can be used to specify the database resources that you want to integrate with. Once you have filled in the discovery options Runbook Studio will provide additional parameters, and in some cases, filters which can be used to configure the activity.

Smart Parameters

Unlike standard command activities whose parameters are determined by the Parameter Set that is selected, the parameters in the Kelverion Integration Module for Microsoft SQL Server are determined by the Discovery options that you specify.

For example, when using the **Insert-SqlRow** activity, the Discovery panel will contain options for selecting Database Name and Table Name. Once you have selected a table, Runbook Studio will provide you with parameters that coincide with the columns in the table schema. If you select another table, Runbook Studio will provide you with a different set of parameters automatically.

You must configure all mandatory parameters. To view the optional parameters that are associated with an activity, click **Optional** at the top of the Parameters tab.

In addition, all activities in the Kelverion Integration Module for Microsoft SQL Server include a **Connection** parameter which is used to specify information that the activity will use to connect to SQL Server when it is executed as part of a runbook running in Azure. Typically, you will assign a Connection Asset data source to this parameter so that the activity can securely use connection information stored in Azure. The Connection parameter should not be confused with the similarly named Connection option on the Discovery panel which is used to specify how Runbook Studio connects to SQL Server to provide design-time configuration options.

Several factors determine the data sources that are available when configuring a parameter. They include: the parameter's data type, whether it is linked to another activity and whether the runbook has any input parameters.

Runbook studio supports the following data sources:

Data Source	Description
Activity output	Specify activity whose output will be assigned to the parameter. You may also provide an optional Path to select a specific property of the output objects that are generated by the activity. Available when the activity is linked to a source activity.
Not configured	Clears any value that was previously configured. You must configure all mandatory parameters.
Certificate asset	Specify the name of the global certificate asset that will be used to provide a value for the parameter. If you have connected to Azure and selected a Subscription and Automation Account on the toolbar, the data source will provide the names of the certificates that are available.
Credential asset	Specify the name of the global credential asset that will be used to provide a value for the parameter. If you have connected to Azure and selected a Subscription and Automation Account on the toolbar, the data source will provide the names of the credentials that are available.
Constant	Specify a constant value to assign to the parameter. Available for parameters that have the following data types: • BigInt • Bit • Char • Date • DateTime • DateTime2 • Decimal • Float • Int • Money • NChar • NText • NVarChar • Real • SmallDateTime • SmallMoney • Text • Time • Xml When assigning a constant DateTime value, Runbook Studio assumes the value is in UTC.
Connection asset	Specify the name of the global connection asset that will be used to provide a value for the parameter. If you have connected to Azure and selected a Subscription and Automation Account on the toolbar, the data source will provide the names of the connections that are available.
Empty string	An empty string will be assigned to the parameter. Available when the parameter is type *System.String*
Null	A null ($null) value will be assigned to the parameter. Available when the parameter type is a reference type.

PowerShell expression	Specify a *simple* PowerShell expression whose output will be assigned to the parameter. You can use variables in the expression to access the output of an activity or a runbook parameter.
Runbook input	Specify the name of the runbook input parameter whose value will be assigned to the parameter. Available when the runbook has one or more input parameters.
Variable asset	Specify the name of the global variable asset that will be used to provide a value for the parameter. If you have connected to Azure and selected a Subscription and Automation Account on the toolbar, the data source will provide the names of the variables that are available.

Smart Filters

Some of the activities in the Kelverion Integration Module for Microsoft SQL Server include a Filters panel which lets you specify filters that can be used to retrieve or modify specific rows in a database table.

If you do not provide any filters, then the activity will retrieve or modify all records in the target table.

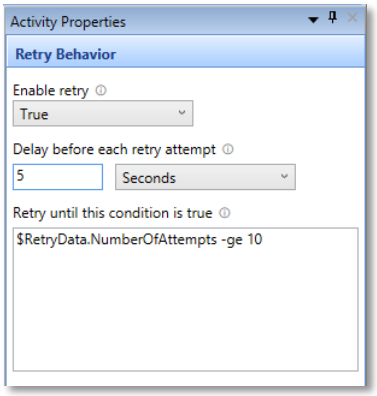
To add a filter to your activity, select the **Filters** panel and click **Add.** Filters have the following properties.

Property	**Description**
Filter	The name of the filter.
Operation	The operation to be used to evaluate the filter. Different operators will be provided based on the filter that is selected. Possible filter operators include: • Equals • Does not equal • Is less than • Is less than or equal to • Is greater than • Is greater than or equal to • Contains • Does not contain • Matches • Does not match • Starts with • Ends with
Value	The data source used to retrieve the value to evaluate the filter. The value used to evaluate the filter will be obtained. For more information on data sources, please refer to the Parameters section for more information on configuring data sources. If you want to filter for a NULL database value, use a PowerShell Expression data

	source and enter the expression **[DBNull]::Value.**

Retry Behavior

The activities in the Kelverion Integration Module for Microsoft SQL Server can be configured to run multiple times until a particular condition, which you specify, is satisfied. You can use the retry behavior options to configure activities that should run multiple times, that are error prone or may need more than one attempt for success.



When you enable retry for an activity, you can configure the runbook to wait a specified number of minutes or seconds before running the activity again. If no delay is specified the runbook will run the activity again, immediately after it finishes running.

The retry condition lets you specify a PowerShell expression that the runbook will evaluate after each time the activity runs. If the result of the expression is true the activity does not run again, and the runbook moves on to the next child activity in the runbook.

When defining the retry conditions for your activity you can take advantage of a global variable called **$RetryData**. Specific information about the last time the activity ran can be accessed using the following properties.

Property	Description
NumberOfAttempts	Number of times that the activity has ran
Output	Output that was generated by the activity the last time that it ran
TotalDuration	Time elapsed since the activity was started
StartedAt	Time in UTC when the activity was first started

The following are some examples of activity retry conditions.

```
# Run the activity exactly 5 times
$RetryData.NumberOfAttempts -eq 5

# Run the activity until it produces some output
$RetryData.Output.Count -ge 1

# Run the activity until at least 2 minutes has elapsed
$RetryData.TotalDuration.TotalMinutes -ge 2
```

Additional Parameters

The activities in the Kelverion Integration Module for Microsoft SQL Server let you specify additional PowerShell parameters that you can use to control the behavior of the activity.

For example, to output detailed information about the operation performed by an activity you would specify **-Verbose:$True.**

Delete-SqlRow

The **Delete-SqlRow** activity is used in a runbook to delete rows from a database table or view.

When deleting rows from a view, the view must be updatable and reference exactly one base table in the FROM clause of the view definition. For more information about updatable view, see CREATE VIEW (Transact-SQL).

Filters are used to control which rows of the target table are to be deleted**. If you do not define any filters, the activity will truncate every row in the table.**

Discovery Parameters

You can use the following discovery options to connect to SQL Server and configure the activity:

Connection	The name of the Smart Connection used to connect Runbook Studio to Microsoft SQL Server.
Database Name	The name of the target database
Include Encrypted Columns	Enable support for filtering on columns encrypted. ***Note:*** only columns that are encrypted with **deterministic** encryption are supported and you are limited to the **Equal to** operator.
Table Name	The name of the database table to delete rows from.

Required Parameters

You must configure the following parameters:

Connection	A hashtable containing connection information. This is typically obtained using a Connection Asset data source or Get-AutomationConnection activity.

Optional Parameters

You can use the following optional parameters to control the behavior of the activity:

Command Timeout	The number of seconds that the activity will wait for the database command to be executed before failing with an error.

Filters

The activity will provide filters based on the columns in the table that you selected. You can configure one or more filters to determine which rows to delete. ***If you do not define any filters, the activity will truncate every row in the table.***

Output

The activity outputs the number of rows that were deleted. When no filters are provided, the activity returns -1 to indicate that the table has been truncated.

Insert-SqlRow

The **Insert-SqlRow** activity is used in a runbook to insert a row into a database table or view.

When inserting into a view, the selected view must be updatable and reference exactly one base table in the FROM clause of the view. For example, an insert into a multi-table view must use only reference columns from one base table. For more information about updatable view, see CREATE VIEW (Transact-SQL).

Discovery Parameters

You can use the following discovery options to connect to SQL Server and configure the activity:

Connection	The name of the Smart Connection used to connect Runbook Studio to Microsoft SQL Server.
Database Name	The name of the target database
Include Encrypted Columns	Enable support for filtering on columns encrypted.
Table Name	The name of the database table into which the row will be inserted

Required Parameters

If the table you selected has columns that do not allow NULL, then the activity will provide parameters that you must configure. You must also configure the following parameters:

Connection	A hashtable containing connection information. This is typically obtained using a Connection Asset data source or Get-AutomationConnection activity.

Optional Parameters

If the table that you selected has any columns that allow NULL, then the activity will provide corresponding parameters that you can configure. You can also use the following parameters to control the behavior of the activity:

Command Timeout	The number of seconds that the activity will wait for the database command to be executed before failing with an error.

Output

If the table that the row is inserted into has an Identity column, then the activity outputs the identity value; otherwise, the activity does not generate any output.

Invoke-SqlProcedure

The **Invoke-SqlProcedure** activity is used in a runbook to invoke a stored procedure.

Discovery Parameters

You can use the following discovery options to connect to SQL Server and configure the activity:

Connection	The name of the Smart Connection used to connect Runbook Studio to Microsoft SQL Server.
Database Name	The name of the target database
Procedure Name	The name of the database procedure to invoke
Execute Mode	Specifies the execution mode. Select **ExecuteQuery** to execute the SQL command and return the rows. Select **ExecuteNonQuery** to execute the SQL command and return the number of rows that were affected. Select **ExecuteScaler** to execute the query and return the first column of the first row in the result set returned by the query (additional columns or rows are ignored).

Required Parameters

You must configure the following parameters:

Connection	A hashtable containing connection information. This is typically obtained using a Connection Asset data source or Get-AutomationConnection activity.

Optional Parameters

You can use the following optional parameters to control the behavior of the activity:

Command Timeout	The number of seconds that the activity will wait for the database command to be executed before failing with an error.

Output

The output returned from the cmdlet is determined by the *Execute Mode* option that you selected.

ExecuteQuery	Zero or more objects, representing the results returned from the SQL query. Each custom object contains properties for each column in the retrieved rows
ExecuteNonQuery	An object that represents the result of the stored procedure and any output parameters.
ExecuteScaler	The first column of the first row in the result set returned by the query (additional columns or rows are ignored).

Invoke-SqlCommand

The **Invoke-SqlCommand** activity is used in a runbook to invoke an SQL query.

Discovery Parameters

You can use the following discovery options to connect to SQL Server and configure the activity:

Connection	The name of the Smart Connection used to connect Runbook Studio to Microsoft SQL Server.
Database Name	The name of the target database
Execute Mode	Specifies the execution mode. Select **ExecuteQuery** to execute the SQL command and return the rows. Select **ExecuteNonQuery** to execute the SQL command and return the number of rows that were affected. Select **ExecuteScaler** to execute the query and return the first column of the first row in the result set returned by the query (additional columns or rows are ignored).

Required Parameters

You must configure the following parameters:

Connection	A hashtable containing connection information. This is typically obtained using a Connection Asset data source or Get-AutomationConnection activity.
Query	The SQL query to execute

Optional Parameters

You can use the following optional parameters to control the behavior of the activity:

Command Timeout	The number of seconds that the activity will wait for the database command to be executed before failing with an error.

Output

The output returned from the cmdlet is determined by the *Execute Mode* option that you selected.

ExecuteQuery	Zero or more custom PSObject objects, representing the results returned from the SQL query. Each custom object contains properties for each column in the retrieved rows
ExecuteNonQuery	The number of rows that were affected by the SQL command.
ExecuteScaler	The first column of the first row in the result set returned by the query (additional columns or rows are ignored).

Select-SqlRow

The **Select-SqlRow** activity is used in a runbook to select rows from a database table or view using filter criteria that you specify.

Discovery Parameters

You can use the following discovery options to connect to SQL Server and configure the activity:

Connection	The name of the Smart Connection used to connect Runbook Studio to Microsoft SQL Server.
Database Name	The name of the target database
Include Encrypted Columns	Enable support for filtering on columns encrypted. ***Note:*** only columns that are encrypted with **deterministic** encryption are supported and you are limited to the **Equal to** operator.
Table/View Name	The name of the target database table or view

Required Parameters

You must configure the following parameters:

Connection	A hashtable containing connection information. This is typically obtained using a Connection Asset data source or Get-AutomationConnection activity.

Optional Parameters

You can use the following optional parameters to control the behavior of the activity:

Command Timeout	The number of seconds that the activity will wait for the database command to be executed before failing with an error.
Order By	Indicates the column that should be used to order the results
Top	Indicates the maximum number or rows to select
Ascending	Indicates whether to order the results in ascending order. The alternative is descending order. Only used when the **Order By** property is defined.

Filters

The activity will provide filters based on the columns in the table that you selected. You can configure one or more filters to determine which rows to select**.**

Output

The activity will output objects that represent the rows that were selected. The properties of the objects are based on the columns in the database table that you selected.

Update-SqlRow

The **Update-SqlRow** activity is used in a runbook to update rows in a database table or view.

When updating rows in a view, the view must be updatable and reference exactly one base table in the FROM clause of the view definition. For more information about updatable view, see CREATE VIEW (Transact-SQL).

Discovery Parameters

You can use the following discovery options to connect to SQL Server and configure the activity:

Connection	The name of the Smart Connection used to connect Runbook Studio to Microsoft SQL Server.
Database Name	The name of the target database
Include Encrypted Columns	Enable support for filtering on columns encrypted. ***Note:*** only columns that are encrypted with **deterministic** encryption are supported and you are limited to the **Equal to** operator.
Table Name	The name of the database table to update

Required Parameters

You must configure the following parameters.

Connection	A hashtable containing connection information. This is typically obtained using a Connection Asset data source or Get-AutomationConnection activity.

Optional Parameters

The activity will provide parameters that correspond to the columns in the database table that you selected. You can use the following parameters to control the behavior of the activity.

Command Timeout	The number of seconds that the activity will wait for the database command to be executed before failing with an error.

Filters

The activity will provide filters based on the columns in the database table that you selected. You can configure one or more filters to determine which rows to update***. If you do not define any filters, the activity will update every row in the table.***

Output

The activity will output the number of records that were updated.

Remarks

If you want to assign a NULL value to a database column, use a PowerShell Expression data source and enter the expression **[DBNull]::Value**.