



# INTEGRATION PACK FOR MICROSOFT SQL SERVER

*For Microsoft System Center Orchestrator*

For System Center 2016 and 2019, you must use the 32-bit version of the integration pack, which has the name **Keverion\_Integration\_Pack\_for\_SqlServer\_3.5**

For System Center 2022 and later, you must use the 64-bit version of the integration pack, which has the name **Keverion\_IP\_SqlServer\_x64\_3.5**

## User Guide

Version 3.5

# Kelverion Integration Pack for SQL Server

Copyright 2012 Kelverion Inc. All rights reserved.

Released: March 2025

*[Feedback](#)*

Send suggestions and comments about this document to [support@kelverion.com](mailto:support@kelverion.com)

# Contents

Introduction .....	4
Installation and Configuration .....	5
System Requirements.....	5
Registering and Deploying the Integration Pack .....	5
Licensing the Integration Pack.....	6
Configuring the Keverion Integration Pack for SQL Server.....	7
Working with Activities in Orchestrator .....	9
Supported SQL Server Data Types.....	9
Common Configuration Instructions for All Activities.....	10
Activity Properties.....	10
General Tab.....	11
Properties/Filters Tab .....	11
Run Behavior Tab .....	13
Published Data .....	13
Delete Rows Activity .....	15
Insert Row Activity .....	16
Monitor Rows Activity.....	17
Run Procedure Activity .....	19
Run Query Activity .....	20
Select Join Activity .....	21
Select Rows Activity .....	23
Update Rows Activity .....	25

# Introduction

---

The Integration Pack for SQL Server is an add-on for Microsoft System Center Orchestrator that lets you query and manipulate SQL Server databases tables.

This Integration Pack was created from experience using Orchestrator in large-scale automation scenarios. The objective was to make the interaction with databases more efficient and easier to use by building more capability into the individual activities and reducing effort to construct and post-process the delimited data, as individual published data items further down the workflow.

When implementing Orchestrator, one of the key best practices for success is to implement a database alongside to provide a persistence store, and audit report and to increase the power and flexibility of the resulting workflows.

When adding a database alongside the Orchestrator there needs to be an efficient connection mechanism. The existing activities create a connection for each transaction and close it after use, even if there is more data to process to the target table. In scenarios of heavy database interaction this puts an unsustainable load on the database access authentication method and for large updates this can increase the workflow run time. An ability to maintain the connection is required to increase performance.

To improve ease of use a key requirement was to remove the need to understand and build SQL statements that would be executed in an activity. Once executed, ideally the activity should exit published data items without any further processing steps or activities. This simplification of workflows would be powerful for both users experienced in Orchestrator workflow development and those new to Orchestrator. The resulting benefits would be to remove any potential errors in the SQL statements, faster workflow development and simpler workflows to maintain and support.

A database is frequently used to track the steps of a process whether it is to do with provisioning, service requests or standard run books. There is a need to be able to flexibly link to the data sources being populated by portals or other request mechanisms. An example here is a service request for a new Virtual Machine within a Virtual Lifecycle Management process or Private Cloud implementation. Typically, this request has multiple steps to perform starting as 'new' and ultimately resulting in 'completed.'

Orchestrator not only needs to monitor for new and updated records but also has different workflows that would need to be triggered at each stage. A flexible monitor activity is required, where the user can define which columns represent the status of the record, either as a numeric or text-based field. This monitor should not only recognize a matching condition to initiate a workflow but also return an appropriate update to that record.

The Kolverion Integration Pack for SQL Server adds these capabilities to the Orchestrator workflow designer and installs them as a compliant Integration Pack. The rest of this guide describes how to install and use the five new activities provided.

# Installation and Configuration

---

The following sections outline how to deploy and configure the Keverion Integration Pack for SQL Server.

## System Requirements

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

### *Keverion\_Integration\_Pack\_for\_SqlServer (32-bit)*

- Microsoft System Center Orchestrator 2016, 2019
- Microsoft .NET Framework 4.7.2

### *Keverion\_IP\_SqlServer\_x64 (64-bit)*

- Microsoft System Center Orchestrator 2022, 2025
- Microsoft .NET Framework 4.7.2

### *One of these database servers:*

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

## Registering and Deploying the Integration Pack

After you download the integration pack file, you must register it with the Orchestrator management server and then deploy it to Runbook Servers and Runbook Designers. For more information about how to install integration packs, see the [How to Install an Integration Pack](https://technet.microsoft.com/en-us/library/hh420346.aspx) (<https://technet.microsoft.com/en-us/library/hh420346.aspx>).

**IMPORTANT:** Ensure that you are deploying the correct version of the Integration Pack.

- For System Center 2016 and 2019, you must use the 32-bit version of the integration pack, which has the name **Keverion\_Integration\_Pack\_for\_SqlServer**
- For System Center 2022 and later, you must use the 64-bit version of the integration pack, which has the name **Keverion\_IP\_SqlServer\_x64**

### *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 later.
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 Keverion 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 Keverion license before running any runbooks that contain activities from the integration pack.

#### *To deploy the integration pack license file to System Center Orchestrator 2019 or earlier:*

1. Copy the **.KAL** license file to %PROGRAMFILES(X86)%\Keverion Automation\Licenses
2. Repeat for each Orchestrator Runbook Server and Runbook Designer host system.

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

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

## Configuring the Kolverion Integration Pack for SQL Server

A configuration establishes a reusable link between Orchestrator and a specific SQL Server database. You can create as many configurations as you require, specifying links to multiple databases. You can also create multiple configurations for the same database to allow for differences in security privileges for different user accounts.

*To set up a SQL Server configuration:*

1. In the Client, click the **Options** menu, and select *KA SQL Server*. The **KA SQL Server** 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. This could be the name of the database or a descriptive name to distinguish the type of configuration.
4. Click the ellipsis button (...) next to the **Type** box and select *SQL Server Options*.
5. In the **Server Name** box, type the name of the SQL Server host. If you are using a non-standard port to connect to SQL Server then append a comma, followed by the port number, to the host name. For example, 192.168.5.23,6283
6. Optionally, in the **Database Name** box, type the name of the database you want to connect to. If you leave the **Database Name** box empty, the activities in the integration will automatically include a cascading **Database Name** property that you can use to select a target database.
7. In the **Authentication Scheme** box, select the method used to authenticate the connection.
8. If *Windows Authentication* is selected, leave the **User Name** and **Password** boxes blank.  
If *SQL Server Authentication* was selected, in the **User Name** and **Password** boxes, type the credentials that Orchestrator will use to connect to the SQL Server database.
9. Optionally, in the **Connection Timeout** box, enter the time in seconds that the Integration Pack will wait for a connection to be established.
10. Optionally, set **Enable Column Encryption** to **True** to enable column-level encryption.
11. Add additional connections if applicable.
12. Click **OK** to close the configuration dialog box, and then click **Finish**.

Authentication Scheme	Behavior
SQL Server Authentication	The IP will connect using the provided User Name and Password values.
Windows Authentication	The IP will connect as the user account configured as the Log On account for the "Orchestrator Runbook Service" on the Orchestrator

	Runbook Server. User Name and Password should be left blank in the IP.
--	--



## Working with Activities in Orchestrator

This integration pack adds the **KA SQL Server** category to the **Activities** pane in the Client. This category contains the following activities:

<b>Delete Rows</b>	Delete rows from a database table or view
<b>Insert Row</b>	Insert a row into a database table or view
<b>Monitor Row</b>	Monitor a database table or view for changes
<b>Run Procedure</b>	Run a stored procedure or function
<b>Run Query</b>	Run an SQL query
<b>Select Join</b>	Select rows from multiple database tables using a join
<b>Select Rows</b>	Select rows from a database table or view
<b>Update Rows</b>	Updates rows in a database table or view

## Supported SQL Server Data Types

The Integration Pack for Microsoft SQL Server supports most SQL Server data types. However, some types, such as *binary* and *varbinary* are excluded, mostly because they cannot be stored in the Orchestrator data bus. Tables that contain columns with unsupported data types can still be accessed by the activities in the integration pack, however columns with unsupported data types will not be available.

*The following data types are supported:*

- bigint
- bit
- char
- date
- datetime2
- datetime
- datetimeoffset
- decimal
- float
- int
- money
- nchar
- ntext
- nvarchar
- numeric
- real
- smalldatetime
- smallint
- smallmoney
- text
- time
- tinyint
- uniqueidentifier
- varchar
- xml

## Working with Date and Time Values

The following table outlines the accepted formats for working with columns that are used to store date and time values.

Column Type	Format	Notes
time	HH:mm:ss	
date	yyyy-MM-ddTHH:mm:ss	Time portion is ignored. Example: 2025-03-04T12:30:15
	yyyy-MM-dd	Example: 2025-03-04
datetime	yyyy-MM-ddTHH:mm:ss	Example: 2025-03-04T12:30:15
datetime2	yyyy-MM-ddTHH:mm:ss	Example: 2025-03-04T12:30:15
smalldatetime	yyyy-MM-ddTHH:mm:ss	Seconds are ignored. Example: 2025-03-04T12:30:15
datetimeoffset	yyyy-MM-ddTHH:mm:ss	Assumes local time. Example: 2025-03-04T12:30:15
	yyyy-MM-ddTHH:mm:ss:ffffffzzz	ISO 8601 round-trip format. Example: 2025-03-04T12:30:15:0000000-5:00
	ddd, dd MM yyyy HH:mm:ss GMT	RFC1123 format. Always UTC. Example: Thu, 06 Mar 2025 12:30:15 GMT
	yyyy-MM-dd HH:mm:ssZ	Universal sortable format. Always UTC. Example: 2025-03-06 12:20:15Z

## 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 activities 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/Filters 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 SQL table.

### *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 SQL Server Connections](#).

## Working with NULL Values

For columns that allow NULL values, you can specify that you want to assign a NULL value to a column for filter by using the **NULL** keyword.

## Filter Behavior

The Select and Monitor activities use filters to determine the values that will invoke a runbook or retrieve activities. Property values of potential candidates are compared to the values of the filters to determine if they meet the criteria. When matching values, you can select one of the available methods of comparison. An option is provided to either match or not match the filter using each method. For example, the "Does not" version of a method finds messages that do not match the filter to start the activity. All text filters are case sensitive.

- **Equals:** the column of the record exactly matches the text or number specified in the filter. Alternatively, you can enter **NULL** as the filter value to perform an *IS NULL* query against the selected column.
- **Does not equal:** the column of the record does not exactly match the text or number specified in the filter. Alternatively, you can enter **NULL** as the filter value to perform an *IS NOT NULL* query against the selected column.
- **Is less than:** the column of the record is less than the number specified in the filter.
- **Is less than or equal to:** the column of the record is less than or equal to the number specified in the filter.
- **Is greater than:** the column of the record is greater than the number specified in the filter.
- **Is greater than or equal to:** the column of the record is greater than or equal to the number specified in the filter.

- **Contains:** the column of the record contains the exact text specified in the filter. Unlike the Equals behavior, there can be other text surrounding the matching text.
- **Does not contain:** the column of the record does not contain the exact text specified in the filter. Unlike the Equals behavior, there can be other text surrounding the matching text.
- **Matches:** the column of the record matches the text specified in the filter. Uses syntax and semantics comparable to the SQL LIKE operator.
- **Does not match:** the column of the record does not match the text specified in the filter. Uses syntax and semantics comparable to the SQL LIKE operator.
- **Starts with:** the column of the record starts with the exact text specified in the filter. Unlike the Equals behavior, there can be other text following the matching text.
- **Ends with:** the column of the record ends with the exact text specified in the filter. Unlike the Equals behavior, there can be other text preceding the matching text.

**Supported wildcards for Matches and Does not match filters.**

Wild Card Character	Description
%	Any string of zero or more characters
_ (underscore)	Any single character
[ ]	Any single character within the specified range ([a-f]) or set ([abcdef])
[^]	Any single character not within the specified range ([^a-f]) or set ([^abcdef])

## Run Behavior Tab

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

### *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 fails.

#### *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#EventNotifications) (https://technet.microsoft.com/en-us/library/hh489610.aspx#EventNotifications).

## Published Data

Published data is the foundation of a working runbook. It is the data produced because 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 subscribe only to data from the activities that are linked to 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 activities, 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. For information about the common published data items, see the [Published Data](http://technet.microsoft.com/en-us/library/hh403821.aspx) (<http://technet.microsoft.com/en-us/library/hh403821.aspx>).

# Delete Rows Activity

The **Delete Rows** activity is used in a runbook to delete rows from a table or view in a Microsoft SQL Server database.

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 \(Transact-SQL\)](#).

## Required Properties

You must configure the following properties:

<b>Database Name</b>	Specifies the name of the target database
<b>Table Name</b>	The name of the database table

## Optional Properties

You can configure the following properties to control the behavior of the activity:

<b>Command Timeout</b>	Specifies 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.**

## Published Data

The activity generates the following published data:

<b>Database Name</b>	Specifies the name of the database that contains the table
<b>Record Count</b>	Specifies the number of records that were deleted or -1 if the table was truncated.
<b>Server Name</b>	Specifies the name of the SQL Server instance
<b>Table Name</b>	Specifies the name of the database table

**Note:** When support for encrypted columns is enabled in the global options, you can create filters that target columns that are encrypted with **deterministic encryption**; however, you are limited to the **Equals** operator.

# Insert Row Activity

---

The **Insert Row** activity is used in a runbook to insert a row into a table a Microsoft SQL Server database.

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 \(Transact-SQL\)](#).

## *Required Properties*

If the table you are inserting rows into has columns that do not allow NULL, then the activity will provide properties that you must configure. You must also configure the following properties:

<b>Database Name</b>	Specifies the name of the target database
<b>Table Name</b>	Specifies the name of the database table to update

## *Optional Properties*

You can configure the following properties to control the behavior of the activity:

<b>Command Timeout</b>	Specifies the number of seconds that the activity will wait for the database command to be executed before failing with an error.
------------------------	---

## *Published Data*

The activity generates the following published data:

<b>Database Name</b>	Specifies the name of the database that contains the table
<b>Server Name</b>	Specifies the name of the SQL Server instance
<b>Table Name</b>	Specifies the name of the database table to update



# Monitor Rows Activity

---

The **Monitor Rows** activity is used in a runbook to monitor a table or view in a Microsoft SQL Server database for changes based on criteria that you specify.

**Important:** The Monitor Rows activity requires the target table to have a column with the Identity property enabled. Also, The Monitor Table activity is designed to work with database tables that have user-defined 'state' columns. State columns are typically integer or bit columns whose values are changed by the system to reflect the status of a table row. As a user you can configure the Monitor Table activity to trigger when the values in these state columns change and then use the optional properties to reset the rows to their un-triggered state.

**Note:** You can use the Monitor Rows activity to monitor views, but only views that are updateable with an identity column. For example, you cannot use Monitor Rows with a view that is based on a join of multiple tables.

## Required Properties

You must configure the following properties:

<b>Database Name</b>	Specifies the name of the target database
<b>Monitor Interval</b>	Specifies the time in seconds between subsequent attempts to poll the database table
<b>Table/View Name</b>	The name of a database table or view

## Optional Properties

The activity will provide optional properties that correspond to the columns in the database table that you selected, and you can use these properties to update the rows that triggered the monitor. You can configure the following properties to control the behavior of the activity:

<b>Ascending Order</b>	Indicates whether to order the results in ascending order. The alternative is descending order.
<b>Command Timeout</b>	Specifies the number of seconds that the activity will wait for the database command to be executed before failing with an error.
<b>Order By</b>	Indicates the column that should be used to order the results
<b>Select Top</b>	Indicates the maximum number of rows to select

## 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 monitor.

## Published Data

The activity generates the following published data:

<b>Database Name</b>	Specifies the name of the database that contains the table or view
<b>Record Count</b>	Specifies the number of records that were updated
<b>Server Name</b>	Specifies the name of the SQL Server instance
<b>Table/View Name</b>	Specifies the name of a database table or view

**Note:** When support for encrypted columns is enabled in the global options, you can create filters that target columns that are encrypted with **deterministic encryption**; however, you are limited to the **Equals** operator.

# Run Procedure Activity

The **Run Procedure** activity is used in a runbook to run a stored procedure or function in a Microsoft SQL Server database.

## Required Properties

You must configure the following properties. If the selected stored procedure or function has any input parameters, the activity will add required parameters to represent them.

<b>Database Name</b>	Specifies the name of the target database
<b>Procedure Name</b>	Specifies the name of the procedure to execute

## Optional Properties

You can configure the following properties to control the behavior of the activity. If the selected stored procedure or function has any output parameters, the activity will add optional parameters to represent them.

<b>Command Timeout</b>	Specifies the number of seconds that the activity will wait for the database command to be executed before failing with an error.
------------------------	---

## Published Data

The activity generates the following published data. If the selected stored procedure or function has any output parameters, the activity will add published data items to represent them.

<b>Database Name</b>	Specifies the name of the target database
<b>Procedure Name</b>	Specifies the name of the procedure that was executed
<b>Record Count</b>	Specifies the number of records that were returned
<b>Records</b>	Comma separated list of data records that were returned from the procedure
<b>Return Value</b>	Specifies the value that was returned from the procedure

**Note:** An error will be displayed if the selected procedure or function has any input, output or return value parameters that have data types that are not supported by Orchestrator.

# Run Query Activity

---

The **Run Query** activity is used in a runbook to run an SQL query for a Microsoft SQL Server database.

## *Required Properties*

You must configure the following properties:

<b>Database Name</b>	Specifies the name of the target database
<b>SQL Query</b>	Specifies the SQL statement to run.

## *Optional Properties*

You can configure the following properties to control the behavior of the activity:

<b>Command Timeout</b>	Specifies the number of seconds that the activity will wait for the database command to be executed before failing with an error.
------------------------	---

## *Published Data*

The activity generates the following published data:

<b>Database Name</b>	Specifies the name of the target database
<b>Record Count</b>	Specifies the number of records that were returned
<b>Records</b>	Comma separated list of data records that were returned from the query
<b>SQL Query</b>	Specifies the name of the procedure that was executed

# Select Join Activity

---

The **Select Join** activity is used in a runbook to select rows from multiple joined tables in a Microsoft SQL Server database using join and filter criteria that you specify.

The Select Join activity is provided for advanced scenarios. Runbook authors should take particular care to ensure that their join conditions are well formed and that the joins that they are building are as simple as possible. Failure to do could impact the performance of your runbooks and potentially exhaust the resources available to your system.

## Required Properties

You must configure the following properties:

<b>Database Name</b>	Specifies the name of the target database
<b>Join (1)</b>	Specifies the first table to join.
<b>Join Condition (1)</b>	condition that will be used to join the table specified in Join (1).
<b>Number of Joins</b>	Specifies the number of tables to join. The default is one.
<b>Table Name</b>	Specifies the name of a database table or view

## Optional Properties

You can configure the following properties to control the behavior of the activity:

<b>Ascending Order</b>	Indicates whether to order the results in ascending order. The alternative is descending order. Only used when the <b>Order By</b> property is defined. The default is true.
<b>Command Timeout</b>	Specifies the number of seconds that the activity will wait for the database command to be executed before failing with an error.
<b>Fetch Next</b>	Specifies the number of rows to return after the <b>Offset</b> has been processed. Can only be used in conjunction with <b>Order By</b> and <b>Offset</b> .
<b>Join (2)</b>	The second table to join. Only available if the <b>Number of Joins</b> property is two or greater.
<b>Join (3)</b>	The third table to join. Only available if the <b>Number of Joins</b> property is three or greater.
<b>Join (4)</b>	The fourth table to join. Only available if the <b>Number of Joins</b> property is four.
<b>Join Condition (2)</b>	The condition that will be used to join the table specified in Join (2). Only available if the <b>Number of Joins</b> property is two or greater.
<b>Join Condition (3)</b>	The condition that will be used to join the table specified in Join (3). Only available if the <b>Number of Joins</b> property is three or greater.
<b>Join Condition (4)</b>	The condition that will be used to join the table specified in Join (4). Only available if the <b>Number of Joins</b> property is four.

<b>Offset</b>	Specifies the number of rows to skip before starting to return rows. Can only be used in conjunction with <b>Order By</b> .
<b>Order By</b>	Specifies the column that should be used to order the results
<b>As Percentage</b>	Indicates how to interpret the <b>Top</b> property. When <b>true</b> the value assigned to <b>Top</b> is interpreted as a percentage. When <b>false</b> , the value assigned to <b>Top</b> is interpreted as the number of rows to return. The default is <b>false</b> .
<b>Select Top</b>	Specifies the maximum number of rows or percentage of rows to retrieve.
<b>With Ties</b>	Indicates how to handle ties for last place when using <b>Top</b> and <b>Order By</b> . When <b>true</b> , two or more rows that tie for last place in the limited results set are returned. When <b>false</b> , only the first tie is returned. The default is <b>false</b> .

### *Filters*

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

### *Published Data*

The activity generates the following published data:

<b>Database Name</b>	Specifies the name of the database that contains the table
<b>Record Count</b>	Specifies the number of records that were returned
<b>Server Name</b>	Specifies the name of the SQL Server instance
<b>Table Name</b>	Specifies the name of a database table

# Select Rows Activity

The **Select Rows** activity is used in a runbook to select rows from a table in a Microsoft SQL Server database using filter criteria that you specify.

## Required Properties

You must configure the following properties:

<b>Database Name</b>	Specifies the name of the target database
<b>Table/View Name</b>	Specifies the name of a database table or view

## Optional Properties

You can configure the following properties to control the behavior of the activity:

<b>Ascending Order</b>	Indicates whether to order the results in ascending order. The alternative is descending order. Only used when the <b>Order By</b> property is defined. The default is <b>true</b> .
<b>Command Timeout</b>	Specifies the number of seconds that the activity will wait for the database command to be executed before failing with an error.
<b>Fetch Next</b>	Specifies the number of rows to return after the <b>Offset</b> has been processed. Can only be used in conjunction with <b>Order By</b> and <b>Offset</b> .
<b>Offset</b>	Specifies the number of rows to skip before starting to return rows. Can only be used in conjunction with <b>Order By</b> .
<b>Order By</b>	Specifies the column that should be used to order the results
<b>As Percentage</b>	Indicates how to interpret the <b>Top</b> property. When <b>true</b> the value assigned to <b>Top</b> is interpreted as a percentage. When <b>false</b> , the value assigned to <b>Top</b> is interpreted as the number of rows to return. The default is <b>false</b> .
<b>Select Top</b>	Specifies the maximum number of rows or percentage of rows to retrieve.
<b>With Ties</b>	Indicates how to handle ties for last place when using <b>Top</b> and <b>Order By</b> . When <b>true</b> , two or more rows that tie for last place in the limited results set are returned. When <b>false</b> , only the first tie is returned. The default is <b>false</b> .

## 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.

## Published Data

The activity generates the following published data:

<b>Database Name</b>	Specifies the name of the database that contains the table
<b>Record Count</b>	Specifies the number of records that were selected

Server Name	Specifies the name of the SQL Server instance
Table/View Name	Specifies the name of a database table or view

**Note:** When support for encrypted columns is enabled in the global options, you can create filters that target columns that are encrypted with **deterministic encryption**; however, you are limited to the **Equals** operator.



# Update Rows Activity

The **Update Table** activity is used in a runbook to update rows in a table or view in a Microsoft SQL Server database.

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 \(Transact-SQL\)](#).

## Required Properties

You must configure the following properties:

<b>Database Name</b>	Specifies the name of the target database
<b>Table/View Name</b>	Specifies the name of a database table or view

## Optional Properties

The activity will provide optional properties 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. You can configure the following properties to control the behavior of the activity:

<b>Command Timeout</b>	Specifies the number of seconds that the activity will wait for the database command to be executed before failing with an error.
------------------------	---

**Note:** To insert a NULL value, use the **NULL** keyword.

## 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 modify.

**Warning:** If you do not define any filters, the Update Table activity will **update every row in the target table/view**.

## Published Data

The activity generates the following published data:

<b>Database Name</b>	Specifies the name of the database that contains the table
<b>Record Count</b>	Specifies the number of records that were updated
<b>Server Name</b>	Specifies the name of the SQL Server instance
<b>Table/View Name</b>	Specifies the name of a database table or view

**Note:** When support for encrypted columns is enabled in the global options, you can create filters that target columns that are encrypted with **deterministic encryption**; however, you are limited to the **Equals** operator.