INTEGRATION MODULE FOR SLACK

For Kelverion Runbook Studio and Azure Automation

USER GUIDE

Version 1.1

Kelverion Integration Module for Slack

Published: December 2021

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 Slack.

System Requirements

The Integration Module for Slack requires the following software to be installed and configured prior to implementing the integration. For more information on installing Kelverion Runbook Studio, please refer to the Kelverion Runbook Studio User Guide.

- Kelverion Runbook Studio 4.7
- Microsoft .NET Framework 4.7.2

Installing the Integration Module

The easiest way to install and deploy the Integration Module for Slack 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 Slack 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 or Hybrid Worker:

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

```
Install-Module -Name Kelverion.Slack -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.

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

Install the Integration Module on your local computer or any Hybrid Workers:

1. Copy the **Kelverion.Slack.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.Slack.zip** file.
5. Copy the **Kelverion.Slack** folder to a location in the `%PsModulePath%` path.

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.Slack.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 the *Kelverion Runbook Studio* and *Automation Connection Assets*.

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

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**.

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 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** 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 ribbon, click the **Home** tab, and then 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 click **Update License Keys**. A summary is displayed.

Authenticating with Slack

Before you can start building runbooks to automate Slack, you must have a Slack App available with the scopes to support the work that you want to do. For example, to post a message using the **Send-SlackMessage** activity you must have a Slack app with the Bot token chat:write scope or User token chat:write, chat:write:user and chat:write:bot scopes.

For more information on creating and configuring Slack Apps, refer to the Authentication section in the Slack API documentation.

To obtain your Slack app's OAuth token:

1. Login to you Slack app.
2. On the left panel, under **Features**, click **OAuth & Permissions**.
3. Scroll down to **OAuth Tokens for your Workspace** and click **Copy** from either the **Bot User OAuth Token** or **User OAuth Token** fields.

Note: You will mostly want to use **Bot tokens** for your runbooks, as they are tied to your application and not to a specific user identity. This will ensure that your runbooks continue to work if the user that installed the Slack app is deactivated.

Azure Global Connection Assets

The activities in the Kelverion Integration Module for Slack require connection information for licensing and to connect to instances of Slack.

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** activity or **Connection Asset** data source.

Adding a global connection asset to your Azure Automation Account:

1. On the **Resources** panel, go to the **Azure** group.
2. Expand the tree to find the desired Azure Subscription and Automation Account.
3. Right click **Connections** and click **Add New Connection.**
4. In the **Name** box, enter a name for the configuration. This could be the name of the instance or a descriptive name to distinguish the type of configuration.
5. In the optional **Description** box, enter a brief description describing the connection.
6. In the **Connection Type** box, select **Kelverion.Slack**.

   In the **AuthToken** box enter the OAuth token used to connect to your workspace. This could be a user token or a Bot token.
7. Click **OK**.

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 Slack.

Activity Properties

All activities in the Kelverion Integration Module for Slack 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 or not 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 activities, 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

The Kelverion Integration Pack for Slack does not support Runbook Studio Smart Discovery.

Activity Parameters

The parameters that are provided by the activities in the Integration Pack for Slack are determined by the parameter set that you select. When you first select an activity, Runbook Studio will automatically display the parameters associated with the default parameter set. Runbook Studio will display all the parameter sets associated with an activity in the **Parameter Sets** pane. When you select a different parameter set, Runbook Studio will automatically update the mandatory and optional parameters to reflect your selection.

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 Slack include a **Connection** parameter which is used to specify information that the activity will use to connect to Slack when it is executed as part of a runbook running in Azure or on a Hybrid Worker. Typically, you will assign a Connection Asset data source to this parameter so that the activity can securely use connection information stored in Azure.

Data Sources

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.

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: • String • DateTime • Timespan • Decimal • Double
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.

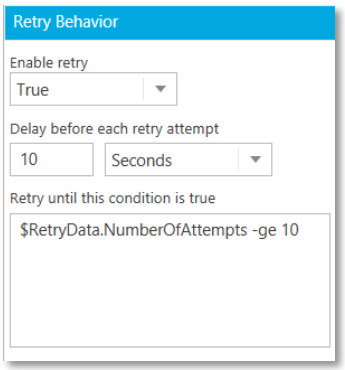
Runbook input	Specify the name of the runbook input parameter whose value will be assigned to the parameter. Available when the runbook has one or more input parameters.
Variable asset	Specify the name of the global variable asset that will be used to provide a value for the parameter. If you have connected 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.

Important: When assigning a constant DateTime and Time values, Runbook Studio assumes the values are in UTC.

Retry Behavior

The activities in the Kelverion Integration Module for Slack can be configured to run multiple times until a 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 completes.



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 Slack 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`**

Rate Limits

The Kelverion Integration Module for Slack is subject to Slack Web API rate limits. When any activity in the integration module detects that Slack rate limits have been exceeded, the activity will fail and with a **Kelverion.Slack.ModelsSlackRateLimitException**.

Activity Reference

The following sections describe how to configure the activities in the Kelverion Integration Module for Slack in conjunction with Kelverion Runbook Studio.

The Integration Module for Slack includes the following activities:

Close-SlackChannel	Archive a channel-based conversation
Close-SlackDirectMessage	Closes a direct message or multi-person direct message
Get-SlackChannel	Gets the channels in a team
Get-SlackChannelHistory	Gets the conversation history for a channel
Get-SlackMessage	Gets details of specific messages
Get-SlackMessageThread	Gets the replies to a message
Get-SlackTeam	Gets information about a team
Get-SlackUser	Gets the users in a team
Get-SlackUserGroup	Gets the user groups for a team
New-SlackChannel	Initiate a public or private channel-based conversation
Open-SlackDirectMessage	Open or resume a direct message or multi-person direct message.
Remove-Message	Remove a message
Send-SlackEphemeralMessage	Send an ephemeral message to a user in a channel
Send-SlackMessage	Send a message to a channel
Send-SlackScheduledMessage	Schedules a message to be send to a channel
Set-SlackMessage	Update a message

Close-SlackChannel

The **Close-SlackChannel** activity archives a channel-based conversation.

Parameter Sets

This activity has the following parameter sets.

```
Close-SlackChannel [-Connection] <Hashtable>
   -ChannelId <String[]>
   [<CommonParameters>]

Close-SlackChannel [-Connection] <Hashtable>
   -Channel <Kelverion.Slack.Models.Channel[]>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

ChannelId	Specifies the ID of the channel to archive.
Channel	Specifies the channel to archive.
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the **Get-AutomationConnection** activity or in graphical runbooks, with a connection asset data source.

Outputs

This activity outputs the ID of the channel that was archived.

Required Scopes

This activity requires **Bot tokens** to have the **channels:manage**, **groups:write**, **im:write** and **mpim:write** scopes. **User** tokens must have the **channels:write**, **groups:write**, **im:write** and **mpim:write** scopes.

Close-SlackDirectMessage

The **Close-SlackDirectMessage** activity closes a direct message or multi-person direct message.

Parameter Sets

This activity has the following parameter sets.

```
Close-SlackDirectMessage
   [-Connection] <Hashtable>
   -ChannelId <String>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

ChannelId	Specifies the ID of the channel to close.
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the **Get-AutomationConnection** activity or in graphical runbooks, with a connection asset data source.

Outputs

This activity outputs **true** if the conversation was closed and **false** to indicate that the conversation was already closed.

Required Scopes

This activity requires Bot tokens to have the **channels:manage**, **groups:write**, **im:write** and **mpim:write** scopes. User tokens must have the **channels:write**, **groups:write**, **im:write** and **mpim:write** scopes.

Get-SlackChannel

The **Get-SlackChannel** activity gets information about the channels in a Slack team.

Parameter Sets

This activity has the following parameter sets.

```
Get-SlackChannel
   [-Connection] <Hashtable>
   -ChannelId <String[]>
   [<CommonParameters>]

Get-SlackChannel
   [-Connection] <Hashtable>
   [-ExcludeArchived <SwitchParameter>]
   [-Limit <Int32>]
   [-TeamId <String>]
   [-MessageTypes {public_channel | private_channel | mpim | im}]
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

ExcludeArchived	Indicates whether to exclude archived channels
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
Limit	Specifies that maximum number of channels to retrieve. By default, the activity will retrieve up to 100 channels.
TeamId	Specifies the encoded team ID to list channels in. Required if the OAuth Token belongs to an org-wide app.
ChannelId	Specifies the ID of channel to retrieve
MessageTypes	Specifies the types of channels to retrieve. Allowed values include **public_channel, private_channel, im** and **mpim**. The default value is **public_channel**. Note that **im** and **mpim** refer to direct message channel and multiparty IM, respectively.

Outputs

This activity generates **Kelverion.Slack.Models.Channel** object that represents the channels the were retrieved from Slack.

Required Scopes

This activity requires Bot and User tokens to have the **channels:read**, **groups:read**, **im:read**, **mpim:read** scopes.

Get-SlackChannelHistory

The **Get-SlackChannelHistory** activity gets the conversation history for a channel-based conversation.

Parameter Sets

This activity has the following parameter sets.

```
Get-SlackChannelHistory
   [-Connection] <Hashtable>
   [-After <DateTimeOffset]>]
   [-Before <[DateTimeOffset]>]
   -ChannelId <String>
   [-Limit <Int32>]
   [<CommonParameters>]

Get-SlackChannelHistory
   [-Connection] <Hashtable>
   -ChannelId <String>
   [-Inclusive <SwitchParameter>]
   [-Limit <Int32>]
   [-Latest <String >]
   [-Oldest <String>]
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

After	Specifies the date and time after which to retrieve messages
Before	Specifies the date and time before which to retrieve messages
ChannelId	Specifies the ID of the channel retrieve the history for
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the **Get-AutomationConnection** activity or in graphical runbooks, with a connection asset data source.
Inclusive	Indicates whether to include messages that occur at the beginning or end of the specified date and time range
Latest	Specifies the timestamp that marks the upper limit of the of the time range from which to retrieve message history.
Limit	Specifies the maximum number of messages to retrieve. The default is 100 messages.

Oldest	Specifies the timestamp that marks the lower limit of the of the time range from which to retrieve message history.

Outputs

This activity generates **Kelverion.Slack.Models.Message** objects that represent the Slack messages that were retrieved.

Required Scopes

This activity requires Bot and User tokens to have the **channels:history**, **groups:history**, **im:history** and **mpim:history** scopes.

Get-SlackMessage

The **Get-SlackMessage** activity gets information about one or more messages.

Parameter Sets

This activity has the following parameter sets.

```
Get-SlackMessage
   [-Connection] <Hashtable>
   -ChannelId <String>
   -Timestamp <String[]>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
ChannelId	Specifies the ID of the channel that contains the messages
Timestamp	The Timestamp(s) of the messages to retrieve

Outputs

This activity generates **Kelverion.Slack.Models.Message** objects that represent the Slack messages that were retrieved.

Required Scopes

This activity requires Bot and User tokens to have the **channels:history**, **groups:history**, **im:history** and **mpim:history** scopes.

Get-SlackMessageThread

The **Get-SlackMessageThread** activity gets the replies to a specified message.

Parameter Sets

This activity has the following parameter sets.

```
Get-SlackMessageThread
   [-Connection] <Hashtable>
   -ChannelId <String>
   -Thread <String>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

ChannelId	Specifies the ID of the channel retrieve the history for.
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
Thread	Specifies the unique identifier of the thread's parent message. The ID must be the timestamp of an existing message with one or more replies. If there are no replies, then just the single message referenced by the timestamp will be retrieved.

Outputs

This activity generates **Kelverion.Slack.Models.Message** objects that represent the Slack messages that were retrieved.

Required Scopes

This activity requires Bot and User tokens to have the **channels:history**, **groups:history**, **im:history** and **mpim:history** scopes.

Get-SlackUser

The **Get-SlackUser** activity gets information about the users in a Slack team.

Parameter Sets

This activity has the following parameter sets.

```
Get-SlackUser
   [-Connection] <Hashtable>
   [-Limit <Int32>]
   -TeamId <String>
   [<CommonParameters>]

Get-SlackUser [-Connection] <Hashtable>
   [-IncludeLocale]
   -UserId <String[]>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the **Get-AutomationConnection** activity or in graphical runbooks, with a connection asset data source.
Limit	The maximum number of items to retrieve.
TeamId	Specifies the ID of the team to retrieve users for. Required if the OAuth Token belongs to an org-wide app.
UserId	Specifies the ID of the user to retrieve information for.

Outputs

This activity generates **Kelverion.Slack.Models.User** that represent the Slack users that were retrieved.

Required Scopes

This activity requires that Bot and User tokens have the **users:read** scope.

Get-SlackUserGroup

The **Get-SlackUserGroup** activity gets information about the user groups in a Slack team.

Parameter Sets

This activity has the following parameter sets.

```
Get-SlackUserGroup [-Connection] <Hashtable>
   [-IncludeDisabled] -TeamId <String>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
IncludeDisabled	Indicates whether to retrieve disabled user groups.
TeamId	Specifies the ID of the team from which to retrieve user groups for.

Outputs

This activity generates **Kelverion.Slack.Models.UserGroup** object that represent the user groups that were retrieved.

Required Scopes

This activity requires Bot and User tokens to have the **usergroups:read** scope.

New-SlackChannel

The **New-SlackChannel** activity initiates a new public or private channel-based conversation.

Parameter Sets

This activity has the following parameter sets.

```
New-SlackChannel [-Connection] <Hashtable>
   -Name <String>
   [-Private]
   [-TeamId <String>]
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
Name	Specifies the name of the channel.
Private	Indicates that the channel should be private.
TeamId	Specifies the encoded team ID to create the channel in. Required if the OAuth Token belongs to an org-wide app.

Outputs

This activity generate a **Kelverion.Slack.Models.Channel** object that represents the channel that was created.

Required Scopes

This activity requires **Bot** and **User** tokens to have the **channels:manage**, **groups.write**, **im:write** and **mpim:write** scopes.

Open-SlackDirectMessage

The **Open-SlackDirectMessage** activity opens or resumes a direct message or multi-person direct message.

Parameter Sets

This activity has the following parameter sets.

```
Open-SlackDirectMessage [-Connection] <Hashtable>
   -ChannelId <String>
   [<CommonParameters>]

Open-SlackDirectMessage [-Connection] <Hashtable>
   -UserId <String[]>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

ChannelId	Specifies the ID of the direct message or multi-person direct message to resume.
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
UserId	Specifies the IDs of the users to include. If only one user is specified, a 1:1 direct message is created.

Outputs

This activity outputs the ID of the channel that was opened or resumed.

Required Scopes

This activity requires **Bot tokens** to have the **channels:manage**, **groups:write**, **im:write** and **mpim:write** scopes. **User** tokens must have the **channels:write**, **groups:write**, **im:write** and **mpim:write** scopes.

Remove-SlackMessage

The **Remove-SlackMessage** activity removes one or more Slack messages.

Parameter Sets

This activity has the following parameter sets.

```
Remove-SlackMessage
   [-Connection] <Hashtable>
   -ChannelId <String>
   -Timestamp <String[]>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

ChannelId	Specifies the ID of the channel that contains the message.
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
Timestamp	Specifies the timestamps of the messages to remove.

Outputs

The activity outputs the timestamps of the messages that were removed.

Required Scopes

This activity requires **Bot tokens** to have the **chat:write** scope. **User** tokens must have the **chat:write**, **chat:write:user** and **chat:write:bot** scopes.

Send-SlackEphemeralMessage

The **Send-SlackEphemeralMessage** activity sends an ephemeral message to a user in a channel. Ephemeral messages are context-sensitive messages, and they are only visible to the specified recipient.

Parameter Sets

This activity has the following parameter sets.

```
Send-SlackEphemeralMessage
   [-Connection] <Hashtable>
   -Channel <String>
   [-Emoji <String>]
   [-Icon <String>]
   -Text <String>
   [-Thread <String>]
   -UserId <String>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

Channel	Specifies the channel, private group, or IM channel to send the message to. Can be encoded ID or a name.
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
Emoji	Specifies the emoji to use as the icon for this message. Overrides the Icon parameter.
Icon	Specifies the URL to an image to use as the icon for this message.
Text	Specifies the text message to send.
Thread	Specifies the timestamp of another message. Avoid using the timestamp of a reply message and instead use the timestamp of the parent message. Ephemeral messages in threads are only show if there is already an active thread.
UserId	Specifies the ID of the user who will receive the ephemeral message. The user should be in the channel specified by the ChannelId parameter.

Outputs

This activity outputs the timestamp of the message that was sent.

Required Scopes

This activity requires **Bot tokens** to have the **chat:write** scope. **User** tokens must have the **chat:write**, **chat:write:user** and **chat:write:bot** scopes.

Send-SlackMessage

The **Send-SlackMessage** activity sends a message to a Slack channel.

Parameter Sets

This activity has the following parameter sets.

```
Send-SlackMessage
   [-Connection] <Hashtable>
   -Channel <String>
   [-Emoji <String>]
   [-Icon <String>]
   [-ReplyBroadcast]
   -Text <String>
   [-Thread <String>]
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

Channel	Specifies the channel, private group, or IM channel to send the message to. Can be encoded ID or a name.
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
Emoji	Specifies the emoji to use as the icon for this message. Overrides the Icon parameter.
Icon	Specifies the URL to an image to use as the icon for this message.
ReplyBroadcast	Indicates whether the reply should be made visible to everyone in the channel or conversation.
Text	Specifies the text message to send.
Thread	Specifies the timestamp of another message. Avoid using the timestamp of a reply message and instead use the timestamp of the parent message.

Outputs

The activity generates a **Kelverion.Slack.Models.Message** object that represents the message that was sent.

Required Scopes

This activity requires **Bot tokens** to have the **chat:write** scope. **User** tokens must have the **chat:write**, **chat:write:user** and **chat:write:bot** scopes.

Send-SlackScheduledMessage

The **Send-SlackScheduleMessage** activity schedule a message to be sent to a Slack channel.

Parameter Sets

This activity has the following parameter sets.

```
Send-SlackScheduledMessage [-Connection] <Hashtable>
   -Channel <String>
   -DateTime <DateTimeOffset>
   [-ReplyBroadcast]
   -Text <String>
   [-Thread <String>]
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

Channel	Specifies the channel, private group, or IM channel to send the message to. Can be encoded ID or a name.
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the Get-AutomationConnection activity or in graphical runbooks, with a connection asset data source.
DateTime	Specifies the date and time when the message should be sent.
ReplyBroadcast	Indicates whether the reply should be made visible to everyone in the channel or conversation.
Text	Specifies the text message to send.
Thread	Specifies the timestamp of another message. Avoid using the timestamp of a reply message and instead use the timestamp of the parent message.

Outputs

This activity outputs the ID of the scheduled message. Not to be confused with message timestamps.

Required Scopes

This activity requires **Bot tokens** to have the **chat:write** scope. **User** tokens must have the **chat:write**, **chat:write:user** and **chat:write:bot** scopes.

Set-SlackMessage

The **Set-SlackMessage** activity updates one or more Slack messages.

Parameter Sets

This activity has the following parameter sets.

```
Set-SlackMessage
   [-Connection] <Hashtable>
   -ChannelId <String>
   [-Text <String>]
   -Timestamp <String[]>
   [<CommonParameters>]
```

Parameters

This activity has the following parameters.

ChannelId	Specifies the ID of the channel containing the message to be updated.
Connection	Specifies a hashtable that contains information used to license the module and to connect to Slack. Connection information is typically retrieved from Azure Automation using the **Get-AutomationConnection** activity or in graphical runbooks, with a connection asset data source.
Text	Specifies new text for the message.
Timestamp	Specifies the timestamps of the messages to be updated.

Outputs

This activity outputs the timestamps of the messages that were updated.

Required Scopes

This activity requires **Bot** tokens to have the **chat:write** scope. **User** tokens must have the **chat:write**, **chat:write:user** and **chat:write:bot** scopes.