Kelverion Automation
Kelverion Runbook Studio

New User Tutorial

Version 2.0
Table of Contents

Overview .................................................................................................................. 3

Pre-Installation Information ................................................................................. 4
Before you begin ....................................................................................................... 4

Connect to Azure Subscription .............................................................................. 5

Connection Configuration ...................................................................................... 6
Connection Assets .................................................................................................. 6
Create the Runbook Studio Connections ............................................................... 9

Creating your first Runbook .................................................................................. 11

Deeper Discovery Example .................................................................................. 28

Appendix A – Creating an Azure SQL instance and Database ......................... 38
Database configuration .......................................................................................... 38
Create an Azure SQL Server .................................................................................. 38
Configure the SQL Firewall to allow client Access .............................................. 39
Create an Azure SQL Database ............................................................................ 41
Import the SQL Script .......................................................................................... 42
Overview

The Kelverion Runbook Studio is a graphical runbook authoring environment, that can be used to generate Graphical PowerShell and Graphical PowerShell Workflow runbooks for Azure Automation, as well as generating PowerShell snippets which can be used within SMA and your own scripts.

Graphical runbook authoring within the Runbook Studio allows you to focus on the logic flow of your automation without getting bogged down with syntax, this change of focus opens the doors for many more people to exploit the power of Azure Automation. To further simplify the design and maintenance of your runbooks, the Runbook Studio provides “Smart Discovery” when used in conjunction with many cmdlets. Smart discovery allows you to use the Runbook Studio GUI to explore the application, cmdlets, and API that you are interacting with and generate runbooks with a few quick clicks.

The Runbook Studio can offer the greatest acceleration to your runbook development when used in conjunction with the Kelverion integration modules. These PowerShell modules can be used in any PowerShell environment and provide a rich integration experience while developing your runbooks, and provide a supported method to integrate into many applications.

This guide will take you through the creation of your first runbook with the Runbook Studio, and show you how to build a graphical runbook and exploit the power of smart discovery.
Pre-Installation Information

Before you begin
This tutorial assumes that you have already configured the following.

1) An Azure subscription.

2) The Runbook Studio is installed and connected to your Azure Subscription.

3) The Kelverion Management Server must be installed and licensed before the Kelverion Integration modules can be used from your Azure environment.

4) The Kelverion Integration Module for SQL Server installed on your machine and imported into your Automation Account. See the integration module user guide for more help.

5) An Azure SQL database called AutomationData that you have executed the attached SQL script against to create the example database table. See Appendix A – Creating an Azure SQL instance and Database. For more help.

For help with the initial installation of the Runbook Studio please see our website(www.kelverion.com) or our YouTube channel (https://www.youtube.com/user/Kelverion)
Connect to Azure Subscription

Once the database is configured we can start configuring the assets within our automation account that will be used within the runbooks. We need to create a connection asset for our database.

When the Runbook Studio starts up it will not be connected to your azure subscription.

If the connection section of the toolbar has all of the buttons grayed out like the image below

Then you need to complete the setup of the runbook studio by following the steps in the user's guide (”REGISTERING RUNBOOK STUDIO WITH AZURE” - Page 6)

Once that is done; Click on “Connect” on the toolbar to start the login process

You will see the familiar Microsoft prompt to login to your azure subscription. Once you have logged in you will notice that the subscription and account dropdowns are populated

If you have multiple automation accounts associated with your subscription, then you can easily manage them all with the Runbook Studio. The selected account is simply the “default” context that you will be working in. You will see all of the accounts and assets that they contain in the “Resources” pane.
You can see here that my “Development” account is currently selected (as it is bold.)

**Connection Configuration**

We must configure the connections that are used at design time and those used at runtime separately.  
The “Connection Assets” within your Azure Automation account are used at the time of runbook execution.  
The “Smart Connections” used within the Runbook Studio are used at runbook design time to discover the properties of the target system and to assist your configuration of the activities.

**Connection Assets**

We will create the connection in our development account.  
Right click on the Connections Item in the tree to bring up the connection management context menu.
As you would expect the right click menu is context sensitive, so clicking at this level just gives us the option to create a new connection. We will see more of these options shortly.

After you click on the “New Connection” option you will see the New Connection Dialogue

Enter

Name: PDS
Connection Type: Kelverion.IntegrationModule.SQLServer
The dialogue will update, as the Runbook Studio knows that more information is required for this type of connection.

The Licenses server URL, and Authorization key are from your management server. If you are using the management server deployment from the Azure Gallery, then the URL will be your cloud service name, and the authorization key can be found in the configuration log. Please see the management server documentation for more information.
This is a good point to try out some of those other context menu items. You can see that if you right click on an existing connection in the “Resources” pane you get different options.

If you open the Azure portal and navigate to the assets within an Automation Account, the changes made in the Runbook Studio are committed directly to Azure.

If you select one of those assets, then you will see the edit blade open which contains the same properties that you set in the Runbook Studio.

Create the Runbook Studio Connections
The Runbook Studio has the capability to “discover” the properties of resources that you are connecting to within your runbooks, this discovery process really accelerates your runbook development, as it takes care of the PowerShell syntax, and parameter naming for those activities that support it. It removes a lot of the annoying typos that we all seem to suffer from when debugging the first iteration of a new script.
To facilitate smart discovery, the Runbook Studio needs to be told about the connections that it can make use of, for example we configure a connection to a database, and the integration module and the Runbook Studio work together to show us the tables we can access, the fields within those tables and the appropriate data types they contain.
Under the hood the integration module takes care of generating the PowerShell, or SQL, or Web Service requests that are required by the API for the application you are working with.
We will now create Runbook Studio connections for our SQL database.

In the Runbook Studio, click on the Connections button on the toolbar.

The “Smart Connection” manager will open up. Click on Add.

Set the name of the connection to be “PDS” and set the connection type to “Kelverion.IntegrationModule.SQLServer”, then Complete the properties for your SQL database.
Click on OK

The smart connections are now in place.

Let’s create a new runbook and drop a couple of activities in place to see the power of smart discovery in action.

Creating your first Runbook
In the Runbook Studio click the “New” button on the toolbar

When you click on the new runbook button you will be prompted to select the type of runbook you want to create.
The runbook studio allows you to create both “Graphical PowerShell runbooks” and “Graphical PowerShell workflow” runbooks. For the majority of use cases you should use “Graphical runbooks”. Both types are broadly the same, with some extra functionality in the workflow variant, at the expense of some increase in complexity, and a fairly significant performance penalty.

We've already made use of some of the components of the Runbook Studio GUI, but let's examine those in a little more detail, the image below has the key area’s highlighted.

The yellow area, the toolbar is already familiar too us and as you would expect it allows quick access to many of the key functions.

We've also taken a quick look already at the blue Resources pane. There are 2 modes for the Resources pane;

1) Managing the resources that exist within azure, that is the assets (variables, connections, credentials, and certificates) along with runbooks.
2) Access to the activities that we will add to our runbooks.
If we click on the “on-premises” category, and the “Azure” category we can toggle between those 2 modes.

You may also have noticed the search box at the top of the Resources pane. In the screen print below you can see that the list of cmdlets has been filtered using the search.

You will see that a new empty runbook is created on the runbook canvas, and over at the right-hand side the runbook properties are displayed.

The Green area in the middle of the screen is the runbook canvas, this is where we will place the activities that go to make up our runbook.

The red context menu shows use commands depending on when and where you right click on an area of the runbook studio.

Finally, the purple area to the right is the properties pane, which is context sensitive, and will show the properties of the activities you are configuring, or the runbook itself. Depending on what you have selected.

**Tip:**
Click an empty area in the runbook canvas to get back to the runbook properties after you have configured an activity.
You should have an empty runbook in the canvas, and the runbook properties open in the properties pane. Enter a name for your new runbook. I’m going to go with a traditional runbook name.

Now, ensure that the Resources pane is in “on premise” mode and in the search box type “sqlrow”

Drag and drop an insert activity, followed by a select activity
Point the “exit port” of the Insert-SqlRows activity and you will notice that the point changes to show that you are about to create a link. Click on the exit port and drag a link to the “input port” of the Select-SqlRows activity.
Both of the activities in the runbook have the spanner icon to show that they are not fully configured. If there are validation errors with the configuration of the activity they will turn red to emphasize the need for corrections. Let’s configure the insert activity first.

Click on the activity to select it. And you will notice that the properties pane is updated.

The default name is fine for us right now. In more complex runbooks you will need to ensure that all activity names are unique. On this tab of the activity properties you can also change the handling of errors and the colour of the activity, again the defaults are fine for our example. Click on the “Discovery” section to start the discovery process. This is where the Runbook Studio starts to do the hard work.
Click on the connection dropdown and you will see 1 connection defined “PDS” Select it. The Runbook Studio will connect to the SQL server and retrieve a list of databases that are accessible to us.

Select the “AutomationData” database, and again you will see the Runbook Studio working out what resources we can use.

Finally, we will see a list of tables that we can insert data into. Select the table “OMS_Events” from the database. You can see in my screen print there are a number of tables that I can access. You may also note that I can type in the box to filter the list.
Once you have selected the OMS_EVENTS table, switch to the “Parameters” section.
So, for this particular table we must select the connection (Azure Connection Asset) that will be used at runtime, and then we can choose from the 30 available fields what ones we want to insert into.

If the table was defined with “mandatory” fields they would be listed with the connection (an “optional field” in the database is one that allows NULL’s). The Runbook Studio is already helping validate your configuration here to ensure that you supply values for all of the fields that do not allow NULL’s.

For example, if I change to a different table you can see that there are a number of mandatory fields, along with some optional fields.

Let us set the connection and then one of the optional fields for the OMS_Events table.
Click on the connection parameter, and then on the dropdown that currently shows “not configured”
Select Connection asset

Then in the following dropdown select the “PDS" asset.

You will notice that the spanner icon is removed from the activity at this point (as all of the mandatory properties are configured, however we haven’t selected any data to insert into the database (so it will insert a record with only the ID property, and the _created date set as the data from those fields come from the database constraints that are defined)

Click on the “optional” button and then a list of all of the optional fields is presented.
You can filter that list using the text box at the top, let’s filter for the “type” field.

If you mouse over the information (i) icon you can learn a little more about the field.

Let’s keep things simple for now and set a constant value.
Click on the dropdown that has “not configured” selected currently and select “constant”

And now enter the string TEST in the text box

You will see that once again the Runbook Studio is validating that we are configuring the activity in a manner that is compatible with the underlying application... that's why it’s called “smart discovery”, because it protects us from doing stuff that’s ... less than smart.

So, that’s the first activity completed. Let’s hook up a similar configuration on the “select-SqlRows” activity.

First open the discovery section.
Select the “PDS” connection.
Then select the “AutomationData” database.
Then select the “dbo.OMS_EVENTS” table.
It should look like
Next select the parameters section and set the connection asset “PDS” that Azure will use.

We don’t need to select any of the optional properties (but feel free to take a look.)

Finally, we will create a “filter” for our select statement. Select the filters section.

Then click on add Filter near the top.
The filter dialogue will appear.

Select the _ID field in the first dropdown (notice that the activity shows you all of the fields that its discovered)  
And “equals” as the operator. 
Finally select “Activity Output” as the value.

Once Activity output is select 2 further dropdowns appear. 
The first allows you to select an activity (that executes before this one) in the runbook, and the second allows you to select a property that has been “published”. 
The insert activity only publishes a single output so we only need to select the insert activity.

Your filter should look like:
Click on OK.
And we’re ready to publish our first runbook to Azure.

On the toolbar click the “Publish Draft” button

Then switch to “Azure mode” in the Resources pane.
If you expand your automation account and the runbook nodes you will see the draft version of the runbook has been published into Azure,

You will notice that the runbook has a yellow asterisk next to it, and that in the screen shot the **HybridTest1** runbook has a green tick mark. This tells us that the **hello-sqlworld** runbook is in draft mode, and the **HybridTest1** runbook is fully published.
It will come as no surprise that right clicking on a runbook here will bring up a context menu.

Let’s test the draft version of the runbook. Click on the Test Draft menu item, and the testing dialogue will open.

Using the testing function, you can execute your runbooks on Azure, or on a hybrid worker if you have one configured. We just want to execute in Azure, so click on the start button. You will see feedback that the runbook is starting, and shortly afterward the status bar will show that the runbooks is running, and then complete.
You will see something like this as the output
If you scroll down, you will notice the “type” column has the value “TEST”

<table>
<thead>
<tr>
<th>SysID</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeCollected</td>
<td></td>
</tr>
<tr>
<td>TimeGenerated</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>TEST</td>
</tr>
<tr>
<td>UserName</td>
<td></td>
</tr>
</tbody>
</table>

Job status: Completed

And finally, if you view the data in SQL Server Management studio you will see the following too.

In our first example runbook, we quickly explored Smart Discovery. Let’s take a deeper dive into discovery. This example assumes that you have a connection to the Adventure Works database. The Adventure Works database can be downloaded from Codeplex ([https://msftdbprodsamples.codeplex.com/releases/view/125550](https://msftdbprodsamples.codeplex.com/releases/view/125550)) or deployed as an option when creating a new database in Azure or if you are using the Kelverion gallery VM then the Adventure Works database is already installed in the SQL Express instance. These instructions assume that you are using the SQL Express instance in the Gallery VM.
In this example, we’re going to create a more complex runbook with outputs from the activities used as inputs to the subsequent ones. Just as we would in the majority of the real runbooks we create. So, let’s give the hard-working engineering department some extra holiday time as they’ve done such a great job with our new products.

**NOTE:**
The SQL express database is not accessible from Azure, so the following examples only describe the design time process, and do not include publishing and executing the runbooks in Azure.

Click on the connections button on the toolbar

![Connections button](image)

Click the Add button on the connection manager and enter your database details

![Connection manager](image)

Click on OK
Click on OK again to close the connection manager

Switch the Resources pane to “on-Premises” so we can find the activities we want.

We’re going to use the Select and update activities so,
Type “Rows” in the search box to display the activities from the SQL Server integration module.

Drag 2 Select activities and an update activity on to the runbook canvas and join them up with pipeline links.

Rename the activities:
- Select-SqlRows -> Select Department
- Select-SqlRows1 -> Select Members
- Update-SqlRows -> Update Members
You will notice that each of the activities are red to indicate that they are not fully configured.

We’ll sort that out now, starting with “Select Department”.
Click on the activity and switch to the discovery tab in the properties pane.

Click on the connection drop down, and then select the “localsql” configuration. The Runbook Studio will discover the available databases. Select the “Adventure Works” Database. The Runbook studio will now discover all of the tables that exist within the database. If you start to type “human resources” in the search box you will see that the table list is filtered dynamically to show the tables that you are interested in.
Select the HumanResources.EmployeeDepartmentHistory Table.

The discovery process has enough information now so that it can give us the appropriate options while we configure the runtime properties on the “parameters” tab.

Switch to the parameters tab in the Properties pane.
Set the connection to be any available connection (we’re not going to execute the runbook from Azure so the runtime connection is unimportant.)

Once the connection is set the activity will no longer be red (as all of the mandatory properties have values) however we want to set a filter, so switch to the filters tab in the properties pane, and click on “Add Filter”. The “Edit Filter condition” dialogue will pop up.

Click on the filter dropdown and you will see a list of the columns that exists within the table.
Select DepartmentID.

In the operations field, you will see that we have text specific operators (contains, like, etc.) If you change the filter column to ModifiedDate and check those operations again you will see that the operators are now specific to Date's and Times. That’s an important feature of the discovery, and part of what makes it Smart discovery. The Runbook Studio in conjunction with the integration modules know how to interrogate the system that you are interacting with and take care of the details so that you don’t have to worry about it. It helps you to get the right low level syntax for whatever the target application is, so you don’t need to be a SQL, or Service-Now Guru. You can focus on the runbook logic instead.

Switch the Filter field back to DepartmentID, use the Equal operations and sent the comparison to a constant “1”

![Edit Filter Condition](image)

Click on OK.

We’re done with the configuration of the “Select Department” activity, and it will “publish” each of the fields onto the pipeline for the engineering department for us to “subscribe” to in the later activities.

Let’s configure the “Select Members” activity to get all of the members of the engineering department.

Configure the discovery within the properties pane to point to the “Adventure Works” database, and the “HumanResources.Employee” table (don’t forget the search box for the table name.)

In the “parameters” tab of the properties pane set the Connection again, and then switch to Filters, and add a new filter.

When the “Edit Filter Condition” dialogue opens select the Filter Column “BusinessEntityID” and “Equals” as the operation.
Now we're going to use data that's output from the previous activity rather than a constant value as we did before.

In the Value field select “Activity output”.

The Activity drop down is added, and this allows you to select the activity that we want to consume output from, select the “select department” activity.

Finally, “field path” input will allow you to select the particular property that you want to use.

If you look at the fields available there and check the able definition you will notice that each of the fields in the database table have been made available to us.
Select the “BusinessEntityID” field in the field path.

When the runbook is executed now we will have published the details of all of the members of department 1 onto the pipeline. Our final activity is going to update the properties of those employees.

Click on the final activity in our runbook “Update Employee” and configure the discovery tab to point at the “employee” table again.

Then in the filters tab set the RowGUID to equal the activity output from the select members, and the field path to the RowGUID
Now switch to the parameters tab and set the runtime connection property.

Finally, we will set the new value for the VacationHours, because we want to perform a calculation as before we update the data we will use a PowerShell expression as the input type. This allows us a huge amount of flexibility, but it means that we cannot use the GUI to do all of the hard work.
Select the optional field “VacationHours” (don’t forget you can type in the search box at the top to filter the list)
And set the input type to be “PowerShell expression”
Finally set the expression to be

```powershell
$ActivityOutput["Select Members"].VacationHours + 16
```
Which is the PowerShell way of subscribing to the “VacationHours” property for the “Select Members” activity. Obviously when you type the PowerShell instead of letting the Runbook Studio do the hard work you need to be careful with the spelling’s and Syntax.

So, our Deeper Discovery example is complete. If you’re adventure works database is accessible from Azure then you can publish he runbook and execute it, and give all those hardworking engineers a well-earned rest.

Hopefully you can appreciate the huge time saving that the smart discovery can make when you’re developing runbooks.
Appendix A – Creating an Azure SQL instance and Database.

The process of creating a new SQL Server and database in your Azure subscription is very quick, however there are a number of steps that need to be completed in the correct order. These instructions take you through a basic deployment that will allow you complete the first tutorial.

Database configuration
To start we will create a new SQL instance in Azure, create a database within it and then add the table that the runbooks use to store the data. This will take about 15 minutes.

Create an Azure SQL Server

Login to the Azure portal and select the SQL Server resource type

![Microsoft Azure](image)

Click on the Add Button
Populate each of the fields as shown above. You will need to enter a server name that is unique to you.
Once all of the fields are populated Click on OK.
Your SQL Instance will be deployed this typically takes less than 5 minutes.

**Configure the SQL Firewall to allow client Access**
In the SQL Servers blade, click on the database that has just been created
In the settings section click on “Firewall”

Click on “Add client IP” to add your current IP address to the list of addresses allowed through the firewall.
Then click on “Save”

You can now close the SQL Server blades.
Create an Azure SQL Database
Once your SQL instance is deployed to Azure you can create a new SQL database

Select the SQL Databases resource type in the Azure portal

Once the Databases blade opens Click on the Add button
Populate the fields in the Add Database blade. The runbooks that we will import later assume that the database name is entered as **AutomationData**, using the same database name will simplify the rest of the process, so you should stick with that for now.

Ensure that you use the resource group and server that were created in the previous step. For the purposes of the tutorial the “basic” pricing tier is recommended as we do not require any of the more advanced features that are available.

**Import the SQL Script**
The Zip file that contains all of the tutorial resources contains a .SQL script that we will use to create the table we required in our database. We will use SQL Server management studio to connect to our database and execute the script. The Azure SQL databases can be managed with older versions of SQL management studio; however, you will get the best experience with the latest version.
Start the SQL Management studio (click start and type SSMS to search)

When the SQL Management Studio starts, it will prompt you for the connection details.

Enter the server name and admin account details you used when you created the Azure SQL server above.

Click on connect. Provided you set your client address on the SQL Server Firewall settings you will be connected. If you did not set the firewall, and you have a current version of the management studio installed, you may be prompted to login to the account that is associated with your subscription, so that the management studio can update the firewall settings for you.

Once you are connected the object explorer (by default the left-hand pane) will show a tree view of the resources that you can access.

Expand Databases, then select the AutomationData database that was created in Azure.

On the menu bar select File > Open > File (or press Ctrl+O)
And open the file "Create Table OMS_EVENTS.sql" that was in the zip file containing the tutorial resources.

When the file has opened, you will be presented with a new query tab that looks like the one below.

Use the execute button on the toolbar or press F5 to execute the query.

Once the query completes you should see the messages tab at the bottom of the query updated with the message "Command(s) completed successfully."

In the object explorer expand
Databases > AutomationData > Tables > dbo.OMS_EVENTS > Columns
And you will see the structure of the table as shown below.

The database configuration is now complete.