Kelverion Automation

Office 365 Application for Azure Automation

Deployment Guide

Version 1.6

Email: info@kelverion.com
Web: www.kelverion.com

1 Table of Contents

2 Overview 3

3 General Configuration Steps 3

3.1 Pre-Installation Information 3
  3.1.1 Kelverion Items Required 3
  3.1.2 Other Products Required 4
3.2 Installation Steps 4
3.3 Persistent Data Store 5
3.4 Configure the Automation Portal 5
  3.4.1 Portal Queries 5
  3.4.2 Portal Offerings 6
3.5 Create an Azure system assigned managed identity for starting runbooks 6
3.6 Load Integration Modules 6
3.7 Configure your Automation account using the Runbook Studio 6
  3.7.1 Design Time (Smart Connections) 6
  3.7.2 Azure Variables 8
  3.7.3 Azure Credentials 11
  3.7.4 Azure Runtime connections 13
3.8 Import Runbooks using the Runbook Studio 15
3.9 Runbook Customisation 18
  3.9.1 High Level Overview 18
3.10 Runbook Process Flow 19
  3.10.1 Data Storage (PDS Design) 19
  3.10.2 Gathering the Data 20
  3.10.3 Transposing the Data 21
  3.10.4 Using the Data 22
  3.10.5 Returning the Results 23
3.11 Logic Apps - Scheduling runbook execution 24
  3.11.1 Gather Request Logic App 24
  3.11.2 Return Status Logic App 28
3.12 Testing 32

2 Overview

Businesses would like to give team leaders and managers access to create their own users and allocate licenses for Office 365 via self-service facilities to reduce the workload on the heavily utilised service desk and front-line support teams. Equally with I.T. Security being in the forefront of everyone's minds it is a challenge that requires careful balancing between providing the right facilities to end users, and not compromising the security or integrity of your configuration.

Therefore, companies are increasingly looking for Automation to improve consistency and supportability, along with the self-service facilities whilst maintaining the security controls. To meet this need we have built the Office 365 Application for Azure Automation.

The Office 365 Application for Azure Automation provides an easy to use interface using the Kelverion Automation Portal, users are provided the rights to access portal services via group membership, but these rights are isolated from Office 365, so it's simple to provide portal facilities without providing rights to directly manage users and licenses.

All the actions against the Office 365 system are carried out by your Azure Automation account, so every change is logged and carried out in a consistent way.

All the data that drives the requests in the automation portal is discovered and maintained by the Azure Automation runbooks, so the automation portal is always in sync with the Office 365 management experience.

The Runbooks have been written using the Runbook Studio authoring application and leverage the integration and smart discovery capabilities provided by the Integration Module for SQL Server. These Integration Modules are also available in the PowerShell Gallery.

3 General Configuration Steps

3.1 Pre-Installation Information

The Office 365 Application package contains the following elements:

- Persistent Data Store (PDS) SQL configuration script
- Office 365 Azure Automation Runbooks
- Azure Automation Service Export

3.1.1 Kelverion Items Required

The application requires the following Kelverion products:

- Kelverion Runbook Studio
- Kelverion Integration Module for SQL Server
- Kelverion Automation Portal

If you do not already have Kelverion Integration Modules, Kelverion Automation Portal, or the Kelverion Runbook Studio they can be downloaded for evaluation from our website.

This guide assumes that you have already installed the Runbook Studio and the Automation Portal. If you have not yet installed those products, then please do so before you continue. Each of the product downloads contains its own documentation to guide you through the initial configuration.

3.1.2 Other Products Required

The two following Microsoft PowerShell modules are required:

- Az.Accounts
- Az.Automation
- Azure AD
- MSOnline
- MicrosoftTeams

These modules are available from the PowerShell Gallery

3.2 Installation Steps

As a guide the steps taken are as followed:

1. Configure the PDS database
2. Import and configure the Portal Service (If using the Kelverion Automation Portal)
3. Configure the Service Offerings (If not using the Kelverion Automation Portal)
4. Create the Azure components
    a. Resource Group
    b. Automation Account
    c. Managed Identity
    d. Load Integration Modules
    e. Import Runbooks
    f. Create Smart Connections
    g. Create Azure Variables
5. Create the Gather Runbook 90-XX (If not using the Kelverion Automation Portal)
6. Create the Return Runbook 97-XX (If not using the Kelverion Automation Portal)
7. Configure the Convert Runbook 95-XX
8. Create the Logic Apps

3.3 Persistent Data Store

The Persistent Data Store or PDS is a SQL Server database that is used by these runbooks to allow all the actions that the runbooks take to be carried out in a robust way. The use of the database at each “step” allows us to design the runbooks such that each runbook is simple and can be considered a discrete unit. In programming terms, it allows the runbooks to be modular.

To best exploit the power and flexibility of Azure, the PDS should be deployed to a SQL instance within your Azure subscription.

We will be using a PDS on Azure using Azure's SQL offerings, rather than building a VM and installing SQL. This allows us to deploy the SQL instance and PDS database quickly and with the minimum of maintenance.

1. Create a new Azure SQL Server. Create the SQL Server in a New Resource Group "Automation", ensure that the "Allow azure services to access server" check box is ticked. This means that ALL Azure resources will be allowed to access the SQL Server through the firewall
2. Give your desktop access to the SQL Server through the firewall
3. Create a new Database "AutomationData". It's important to use this name for the PDS, as it is held within the runbooks. The **BASIC** tier is ample performance for testing and evaluation of the application. As it is trivial to scale up or down the databases this should also be the starting point for your deployments unless you know that there is going to be a high volume of alerts right from the outset.
4. Connect to the database using SQL Management Studio
5. Execute the SQL script (*PDS_O365.sql*) from the package to Create the database tables and views.

3.4 Configure the Automation Portal

Import the service request definition Services_O365.export

3.4.1 Portal Queries

The following queries must be updated to point to the customers PDS

- AzureAd.v_enabledUsers
- AzureAd.v_disabledUsers
- AzureAd.v_deletedUsers
- AzureAd.v_CurrentUsers
- AzureAD.v_AvailableSKUs
- AzureAD.v_EmailUsers
- AzureAD.v_LicensedUsers
- AzureAD.v_User_Service_Status
- AzureAD.v_ProvisioningStatus
- AzureAd.v_Groups
- AzureAd.v_Teams
- AzureAd.v_TeamsOut
- AzureAd.v_TeamMembers
- AzureAd.v_TeamSecPolicy

- AzureAD.GroupMembers
- AzureAD.AssignedGroups
- AzureAD.AvailableGroups
- AzureAD.v_mailboxNotOnLitigationHold
- AzureAD.v_mailboxOnLitigationHold

The SQL instance AND authentication details will need to be updated for each query.

3.4.2 Portal Offerings

Check the AD Group matches the correct AD Security Group and change accordingly.
e.g. Default offering AD group is Test_Portal_Users.

3.5 Create an Azure system assigned managed identity for starting runbooks

In the automation account, go to Identity. Enable the System assigned managed identity.

Allocate the role of “Automation Operator” for the resource group that the automation account is in.

3.6 Load Integration Modules

The Integration modules will need to be installed in the following locations

- The Automation Account ("Assets" > "Modules")
- The machine where you are running the Runbook Studio.

The modules can easily be installed on the machine from the PowerShell Gallery.

Visit https://docs.microsoft.com/en-us/azure/automation/automation-update-azure-modules for more information about loading Integration modules into your Automation Account.

The following modules are required

- Kelverion.SqlServer
- Az.Accounts
- Az.Automation
- AzureAD
- MSOnline
- MicrosoftTeams

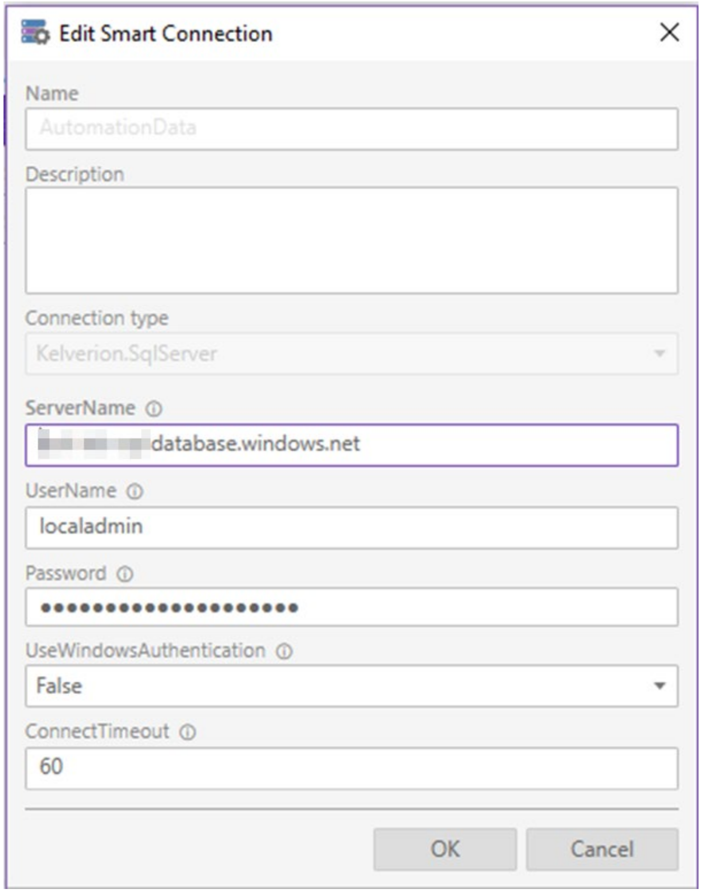
3.7 Configure your Automation account using the Runbook Studio

Connect the Runbook studio to your target Azure subscription. You can find more information on the initial configuration of the runbook studio on pages 5 and 6 of the User’s Guide.

3.7.1 Design Time (Smart Connections)

The runbook studio needs to have connections configured for use at design time, these smart connections are used by the discovery process within the Runbook Studio to allow the Runbook Studio to discover information about your target systems. This discovery process accelerates the process of runbook development.

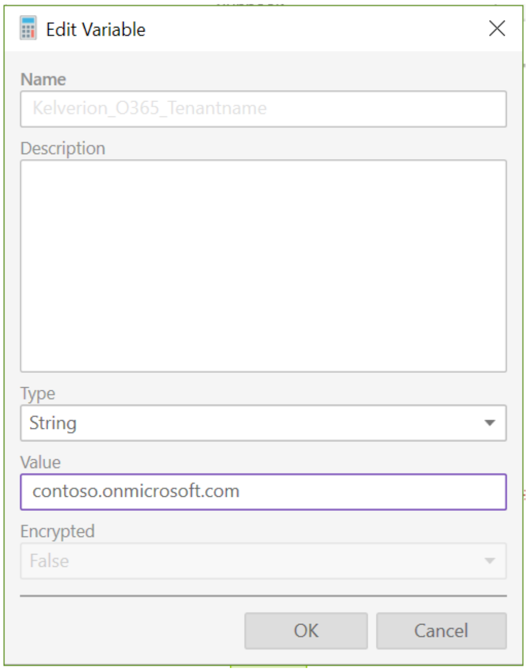
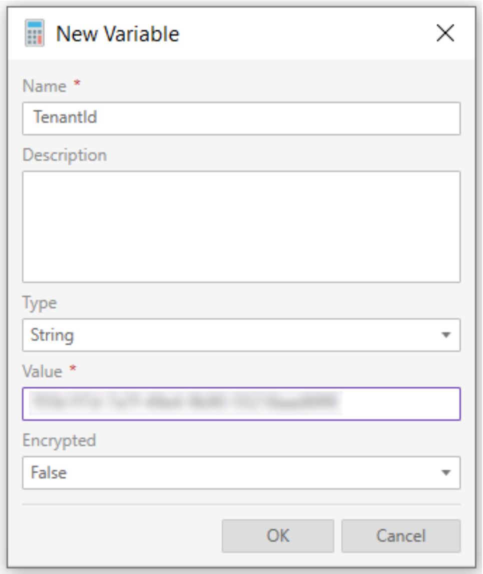
Create the following smart connections within the Runbook Studio.

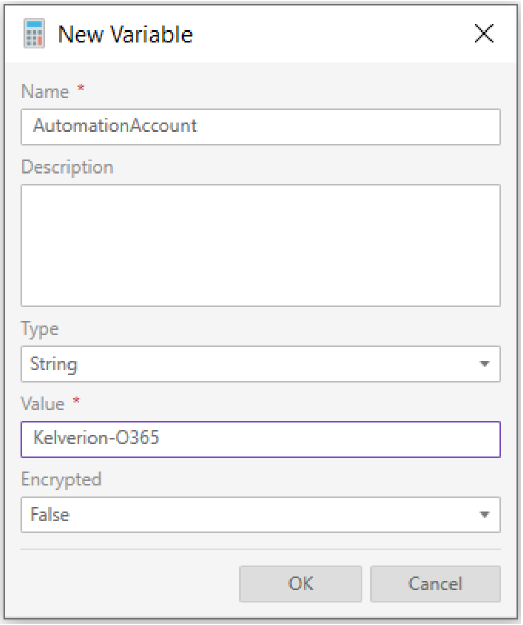
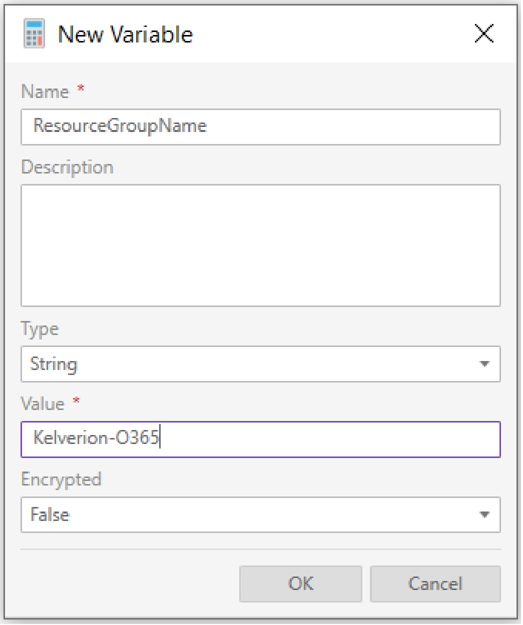
AutomationData	 Edit Smart Connection Name AutomationData Description Connection type Kelverion.SqlServer ServerName [redacted]database.windows.net UserName localadmin Password ●●●●●●●●●●●●●●●●●●●● UseWindowsAuthentication False ConnectTimeout 60

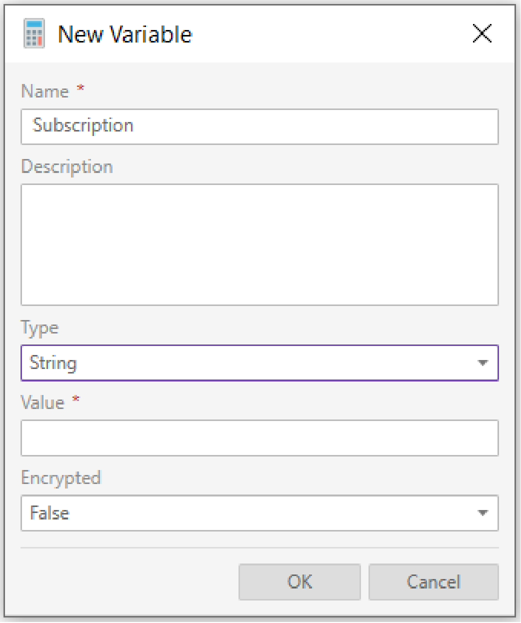
3.7.2 Azure Variables

Azure variables allow us to remove static configuration information from the code in our runbooks and store the information in an easy to access place. This helps us to build consistent configuration between all our runbooks. These values are accessed at design time, and at runtime by both the Hybrid workers, and the Azure Worker sandboxes.

Create the following variables in the Automation Account using the Runbook Studio.

Kelverion_O365_Tenantname	This is the name of your office 365 tenant e.g. contoso.onmicrosoft.com	 Edit Variable Name: Kelverion_O365_Tenantname Description: Type: String Value: contoso.onmicrosoft.com Encrypted: False
TenantId	The Tenant Id of your Azure Subscrption	 New Variable Name *: TenantId Description: Type: String Value *: [redacted] Encrypted: False

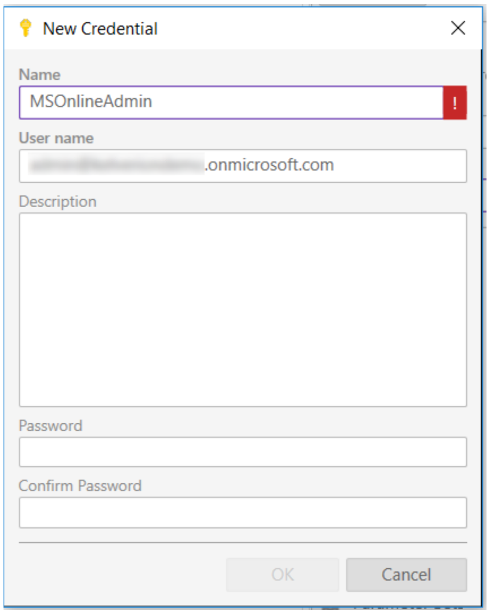
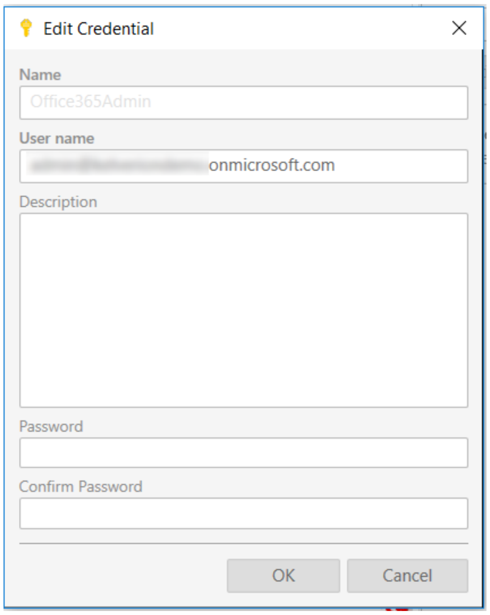
AutomationAccount	The name of the automation account to run the child runbooks	 New Variable Name * AutomationAccount Description Type String Value * Kelverion-O365 Encrypted False
ResourceGroupName	The name of the resource group that has the runbooks	 New Variable Name * ResourceGroupName Description Type String Value * Kelverion-O365 Encrypted False

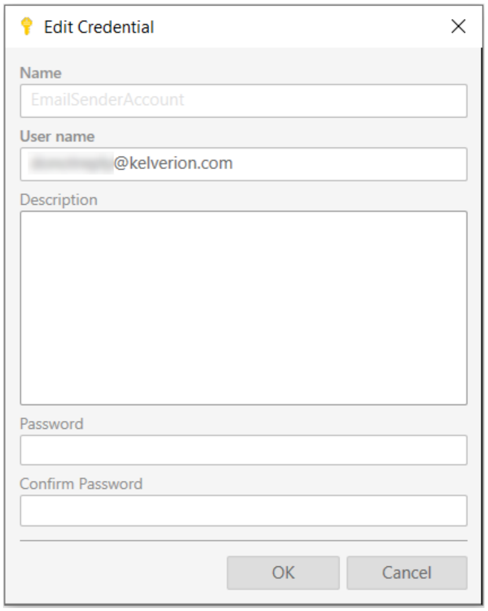
Subscription	The name of the Subscription that you are actioning the runbooks in	 New Variable Name * Subscription Description Type String Value * Encrypted False

3.7.3 Azure Credentials

Azure Credential assets allow us to create and manage credential objects that can be utilised throughout our runbooks. These credentials are accessed by our runbooks at runtime.

Create the following Credential Assets using the Runbook Studio.

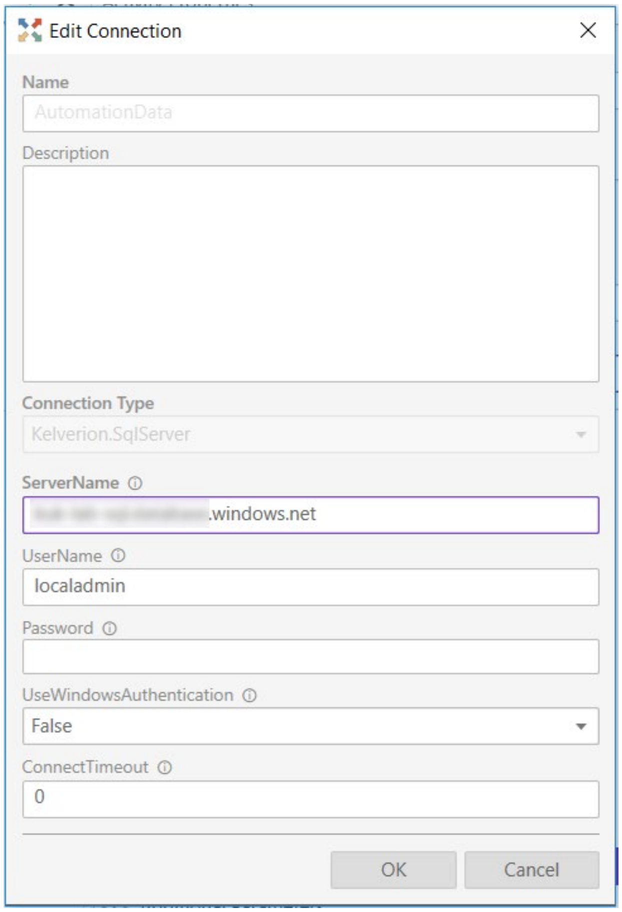
MSOnlineAdmin	This credential will be used to configure the MS Online accounts. This will be an Office 365 administrator account.	 New Credential Name MSOnlineAdmin User name [redacted].onmicrosoft.com Description Password Confirm Password
Office365Admin	This credential will be used to configure the Office 365 users \ groups. Will need to be an Office 365 Administrator	 Edit Credential Name Office365Admin User name [redacted]onmicrosoft.com Description Password Confirm Password

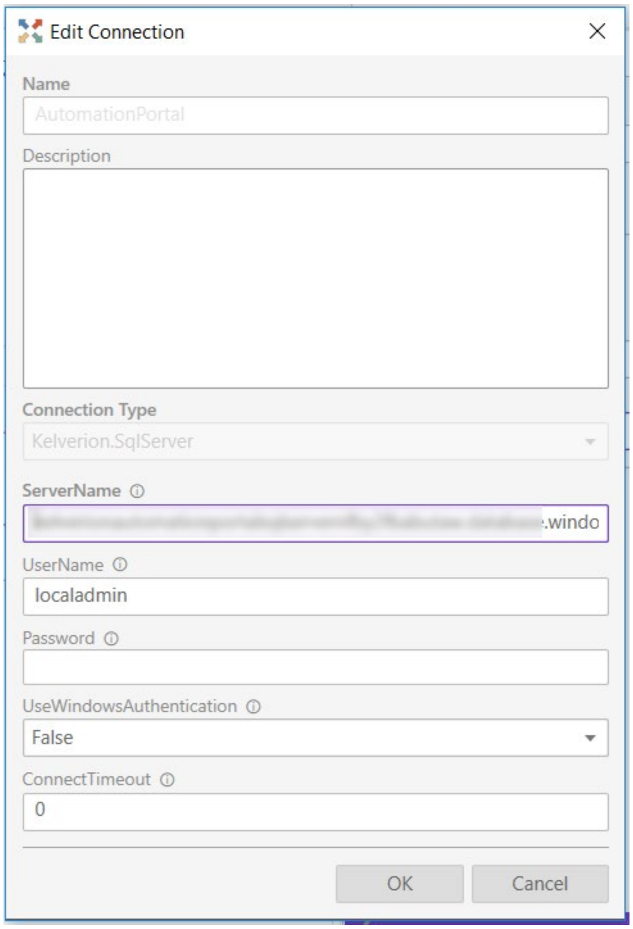
EmailSenderAccount	This credential will be used to email temporary passwords for the “Reset Password” offering.	 Name: EmailSenderAccount User name: [redacted]@kelverion.com Edit Credential Description Password Confirm Password OK Cancel

3.7.4 Azure Runtime connections

Azure connection assets are used at runtime to define a reusable connection configuration. The connection types available are dependent upon module that have been loaded into your Automation Account. If you cannot see the connection types listed below when you attempt to create the connection assets, then this indicates an issue with the modules that are loaded into your automation account. Please verify that all the required modules are loaded.

Create the following Connection in Azure using the Runbook Studio.

AutomationData	The "AutomationData" connection asset in Azure defines the connection that will be used at runtime for the runbooks to connect to the PDS database.	 Name: AutomationData Description: Connection Type: Kelverion.SqlServer ServerName: [redacted].windows.net UserName: localadmin Password: UseWindowsAuthentication: False ConnectTimeout: 0

AutomationPortal	The "AutomationPortal" connection asset in Azure defines the connection that will be used at runtime for the runbooks to connect to the Kelverion Automation Portal Database.	 Edit Connection Name: AutomationPortal Description: Connection Type: Kelverion.SqlServer ServerName: [redacted].windo UserName: localadmin Password: UseWindowsAuthentication: False ConnectTimeout: 0

3.8 Import Runbooks using the Runbook Studio

Connect to the Automation Account using the Runbook Studio.

Check that you have the correct Automation Account set as the default target (if there is more than one Automation Account associated with the subscription)

Open each of the following runbooks, then "publish draft" and then "publish" the runbooks.

Runbook Name	Brief Description
Kelverion_O365_10-1_Worker-Restore-User	This will restore a user account. The account must have been deleted to be able to be restored.
Kelverion_O365_10-2_Worker-Enable-User	This runbook enables an account and allows it to login
Kelverion_O365_10-3_Worker-Delete-User	This runbook deletes the selected account
Kelverion_O365_10-4_Worker-Disable-User	This will disable an account and prevent it from
Kelverion_O365_10-5_Worker-Assign-License	This runbook will assign a license to a user account
Kelverion_O365_10-6_Worker-Unassign-License	This runbook will remove a license from a user account
Kelverion_O365_10-7_Worker-SetLocation	This sets the Office 365 region for an account
Kelverion_O365_10-8_Worker-Modify-User-Service-Availability	This runbook applies which services are available for a user account
Kelverion_O365_10-9_Create-Group	This runbook will create a group
Kelverion_O365_10-10_Add-Group-Members	This runbook will add one or more users to a selected group
Kelverion_O365_10-11_Remove-Group-Members	This runbook will remove one or more users from a selected group
Kelverion_O365_10-12_Add-User-ToGroups	This runbook will add a single user to multiple groups
Kelverion_O365_10-13_Remove-User-FromGroups	This runbook will remove a single user from multiple groups

Kelverion_O365_10-14_Enable-Litigation	This enables email litigation on a user account
Kelverion_O365_10-15_Disable-Litigation	This removes email litigation on a user account
Kelverion_O365_10-16_Create-User	This will create a new user
Kelverion_O365_10-17_Reset-Password	This will reset a user password
Kelverion_O365_10-18_Create-Distribution-Group	This will create an Exchange Online Distribution List. Offering options: Name \ Description \ Owner(s) \ Member(s) \ Moderators(s) (optional)
Kelverion_O365_10-19_Create-Shared-Mailbox	This will create an Exchange Online Shared Mailbox. Offering options: Name \ Full Access \ SendAs. Hardcoded in the runbook code: Quota sizes
Kelverion_O365_20-0_Util-GetPortalData	This runbook will gather Request data from the Automation Portal
Kelverion_O365_30-0_PDSData-Master	This runbook runs all PDS data gathering runbooks
Kelverion_O365_30-1_PDSData-Groups	Gathers data for Groups and populates the Groups table in the PDS
Kelverion_O365_30-2_PDSData-Users	Gathers data for Users and populates the Users table in the PDS
Kelverion_O365_30-3_PDSData-SKUs	Gathers data for SKU and populates the SKU table in the PDS
Kelverion_O365_30-4_PDSData-GroupMembers	Gathers data for Group Members and populates the Group Membership table in the PDS
Kelverion_O365_30-5_PDSData-MSOLUser	Gathers data for MSOLUsers and populates the MSOLUsers table in the PDS
Kelverion_O365_30-6_PDSData-MSOLDeletedUser	Gathers data for deleted MSOLUsers and populates the MSOLUsers table in the PDS
Kelverion_O365_30-7_PDSData-Devices	Gathers data for Azure AD Computers and populates the Computers table in the PDS
Kelverion_O365_30-8_PDSData-Mailbox	Gathers data for Users Mailbox and populates the Mailbox table in the PDS
Kelverion_O365_30-9_PDSData-Teams_Add	Gathers data from Azure AD Teams and populates the Teams table in the PDS

Kelverion_O365_30-10_PDSData-Teams_Remove	Gathers data from Azure AD Teams and the PDS. It runs a comparison and removes redundant data from the PDS
Kelverion_O365_30-11_PDSData-TeamMembers	Gathers data from Azure AD for Team members. It populates the PDS table. It also then runs a comparison and removes any users from the PDS that are not reported from Azure AD
Kelverion_O365_30-12_ExistIn-TeamsMembership	This runbook runs a check against the PDS Team Members table to see if the entries are there
Kelverion_O365_40-01_CreateTeam	Creates a Team
Kelverion_O365_40-02_RemoveTeam	Removes a Team
Kelverion_O365_40-03_AddTeamMember	Adds a member(s) to a Team
Kelverion_O365_40-04_RemoveTeamMember	Removes a member(s) from a Team
Kelverion_O365_40-05_ApplyTeamSecPolicy	Applies a pre-defined security policy to a Team. This is defined in the TeamsSecurityPolicy table in the PDS
Kelverion_O365_90-01_KAP_Get_Request	Gathers the request data from the Kelverion Automation Portal. Requires a Logic App to trigger
Kelverion_O365_95-01_ConvertToRequestData	Converts the gathered data to the correct format for the runbooks
Kelverion_O365_97-01_KAP_Return_Status	Collects processed rows of the PDS for returning data back to the Kelverion Automation Portal. Requires a Logic App to trigger.

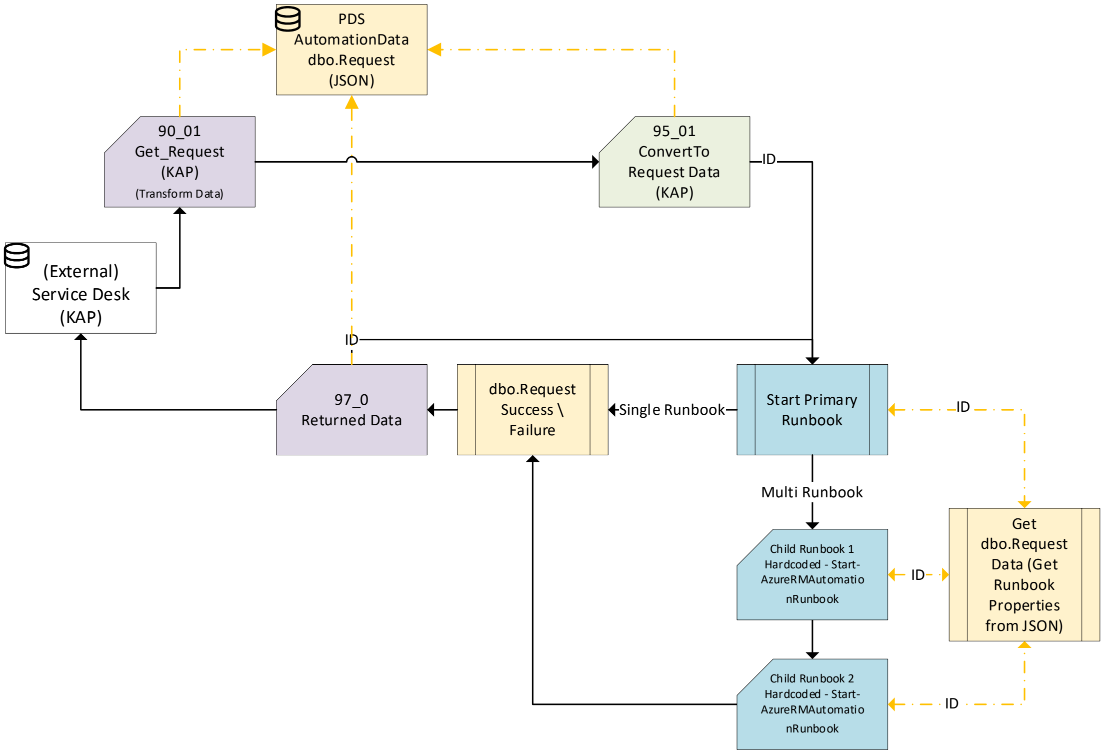
3.9 Runbook Customisation

This application is designed to allow flexibility across multiple service desk applications. The initial runbooks are configured for using the Kelverion Automation Portal as the main user portal for initiating any requests and receiving the status updates.

If you are not using the Kelverion Automation Portal, then the installation engineer will be required to create a gathering runbook (90_XX) and a return runbook (97_XX).

3.9.1 High Level Overview

The main process flow is as follows:

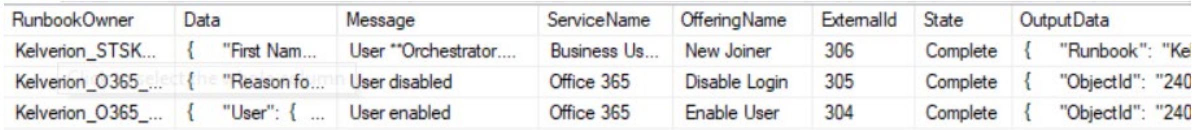


3.10 Runbook Process Flow

This section covers a more detailed description of how the runbook logic fits together. You should be able to use it to configure and customise the application.

3.10.1 Data Storage (PDS Design)

The application uses a SQL database (PDS) to store the request \ offerings from the Service Desk. The table dbo.Request stores data per request from the source Service Desk.
Each row has a unique ID that is used as a reference for the worker runbooks.



RunbookOwner	Data	Message	ServiceName	OfferingName	ExternalId	State	OutputData
Kelverion_STSK...	{ "First Nam...	User "Orchestrator....	Business Us...	New Joiner	306	Complete	{ "Runbook": "Ke
Kelverion_O365_...	{ "Reason fo...	User disabled	Office 365	Disable Login	305	Complete	{ "ObjectId": "240
Kelverion_O365_...	{ "User": { ...	User enabled	Office 365	Enable User	304	Complete	{ "ObjectId": "240

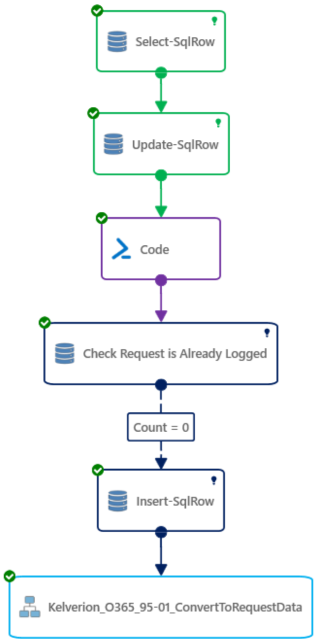
3.10.1.1 Request Table

Description of the column use in the dbo.Request table.

Column	Description
`ID`	Unique column ID. Created when data is inserted.
`Created`	Create time stamp
`Updated`	Updated time stamp
`RequestedBy`	Service Offering Requestor
`Data`	JSON Data dump from the request offering
`Message`	Return message to the service desk
`Deleted`	(Bit) 1 = request deleted
`ServiceID`	Numerical ID of the Service Name (Optional and service desk dependent)
`ServiceName`	Service Name from the service desk request
`OfferingID`	Numerical ID of the Offering Name (Optional and service desk dependent)
`OfferingName`	Offering Name from the service desk request
`ExternalId`	Service Desk reference ID
`State`	Worker Runbook state
`RowVersion`	SQL Row version. Generated automatically on data entry
`ServiceDesk`	Name of the service desk being used
`OutputData`	JSON Data used by the worker runbooks

3.10.2 Gathering the Data

The base application comes with a runbook designed to use the Kelverion Automation Portal. Other Service Desk portals, with available Kelverion Integration Modules, can be used too; but a gathering runbook will need to be created.


Select-SqlRow
Update-SqlRow
Code
Check Request is Already Logged
Count = 0
Insert-SqlRow
Kelverion_O365_95-01_ConvertToRequestData

The runbooks are driven from an Azure Logic App, to either:

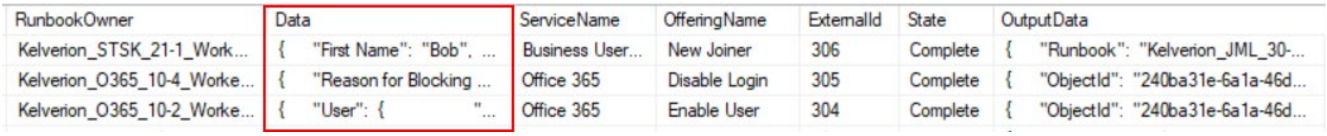
- Monitor the Service Desk portals database (SQL)

or

- Launch the gather runbook every 'x' minute(s)

Each application has a 90-0x_XXX_Get_Request runbook that can be configured for the required Service Desk portal.

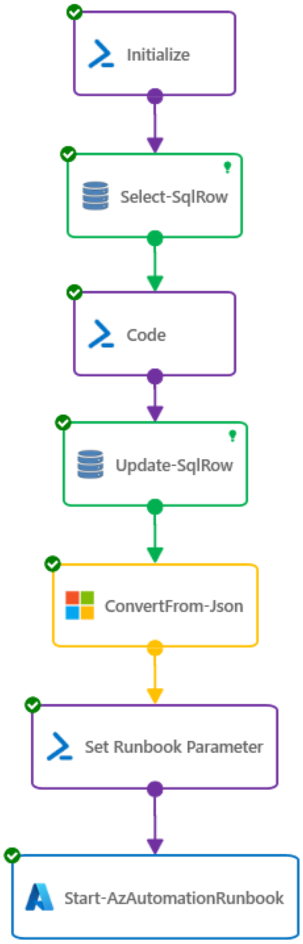
This runbook has a code block that will transpose the Automation Portal request into JSON format and store it into the PDS dbo.Request table. Other service desks will require different activities to store the code in the corresponding JSON format.



RunbookOwner	Data	ServiceName	OfferingName	ExternalId	State	OutputData
Kelverion_STSK_21-1_Work...	{ "First Name": "Bob", ...	Business User...	New Joiner	306	Complete	{ "Runbook": "Kelverion_JML_30-...
Kelverion_O365_10-4_Worke...	{ "Reason for Blocking ...	Office 365	Disable Login	305	Complete	{ "ObjectId": "240ba31e-6a1a-46d...
Kelverion_O365_10-2_Worke...	{ "User": { "...	Office 365	Enable User	304	Complete	{ "ObjectId": "240ba31e-6a1a-46d...

3.10.3 Transposing the Data

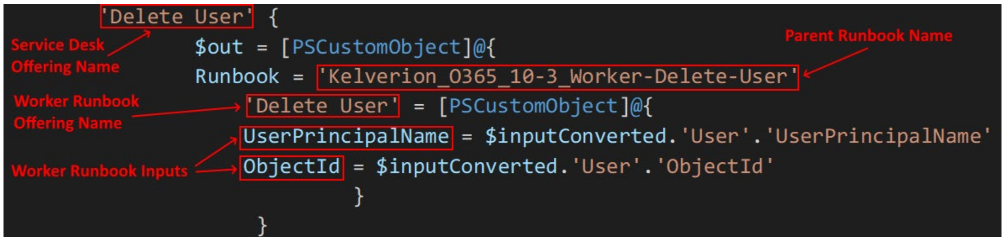
The modular runbooks are designed to use JSON.
However, they require the JSON to be in a specific format. Each application has a runbook (Kelverion_XXX_95-01_ConvertToRequestData) to transpose the data into the required format for the worker runbooks.



These runbooks will all look similar. This depends on if the worker runbooks require the use of a Hybrid Worker. In this case you may see additional logic at the end of the Runbook.

The only activity that will require modification is the “Code” activity. This PowerShell activity has an input of JSON (Data) and transposes it to JSON (OutputData).

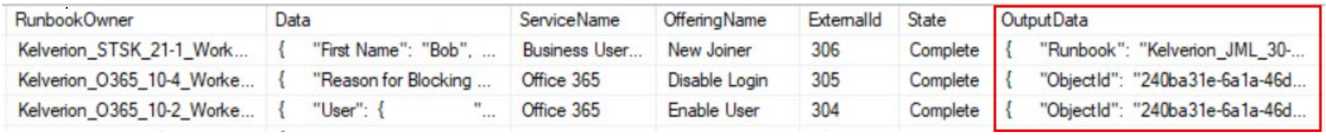
Example Code for an offering with a single worker runbook.



```
        'Delete User' {
                $out = [PSCustomObject]@{
                Runbook = 'Kelverion_O365_10-3_Worker-Delete-User'
                        'Delete User' = [PSCustomObject]@{
                        UserPrincipalName = $inputConverted.'User'.'UserPrincipalName'
                        ObjectId = $inputConverted.'User'.'ObjectId'
                                }
                        }
```

Each service desk offering must pass through the name of the ‘Parent Worker Runbook’. This can be a hidden field in the service desk portal.

The activity Set Runbook Parameter passes the ID column, from the Request table, and feeds it into the Start-AzureRMAutomationRunbook activity.
Start-AzureRMAutomationRunbook launches the Parent Worker Runbook defined from the Service Desk offering.



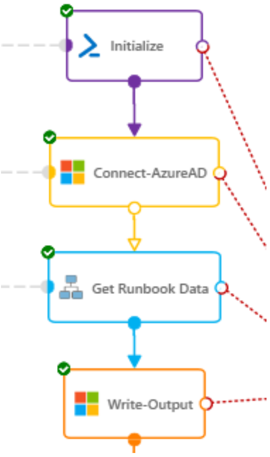
RunbookOwner	Data	ServiceName	OfferingName	ExternalId	State	OutputData
Kelverion_STSK_21-1_Work...	{ "First Name": "Bob", ...	Business User...	New Joiner	306	Complete	{ "Runbook": "Kelverion_JML_30-...
Kelverion_O365_10-4_Worke...	{ "Reason for Blocking ...	Office 365	Disable Login	305	Complete	{ "ObjectId": "240ba31e-6a1a-46d...
Kelverion_O365_10-2_Worke...	{ "User": { "...	Office 365	Enable User	304	Complete	{ "ObjectId": "240ba31e-6a1a-46d...

3.10.4 Using the Data

Each worker runbook has been designed with the following:

- Single input of ID (from the dbo.Request table)
- Calls a child runbook Kelverion_XXXX_20-0_Util-GetRunbookData to gather the required runbook inputs

This allows the correct input data to be gathered back from the PDS.



Here is an example start of a worker runbook.
It will:

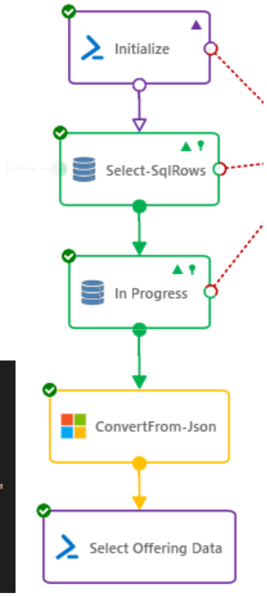
- Set some initial variables (Initialize)
- Connect to Azure (Connect-AzureAD)
- Call ‘Get Runbook Data’ runbook
- Output the returned data (Write-Output) for use in the main runbook activities

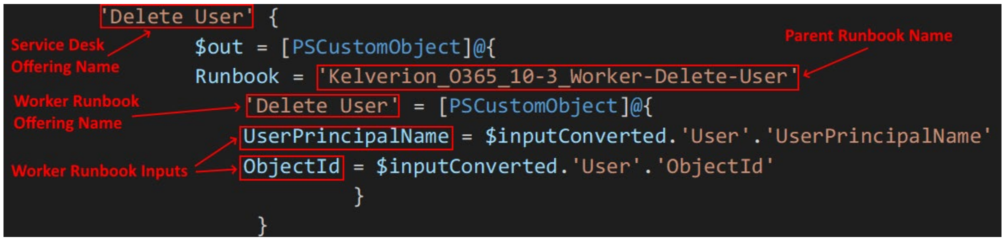
N.B. More complex array outputs may require a PowerShell code block to process them, rather than a Write-Output.

‘Get Runbook Data’ has 2 required inputs:

1. ID
2. OfferingName (Worker Runbook Offering Name)

The activity ‘Get Runbook Data’ uses the ID input to go and retrieve the OutputData from the PDS.
The data is converted back from JSON and the ‘Select Offering Data’ filters out the required data based on the worker runbook offering name.

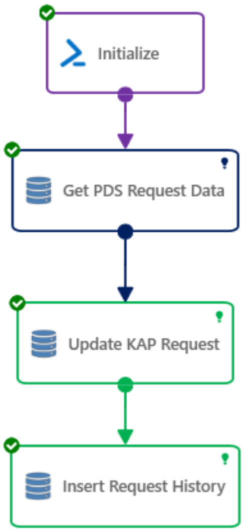


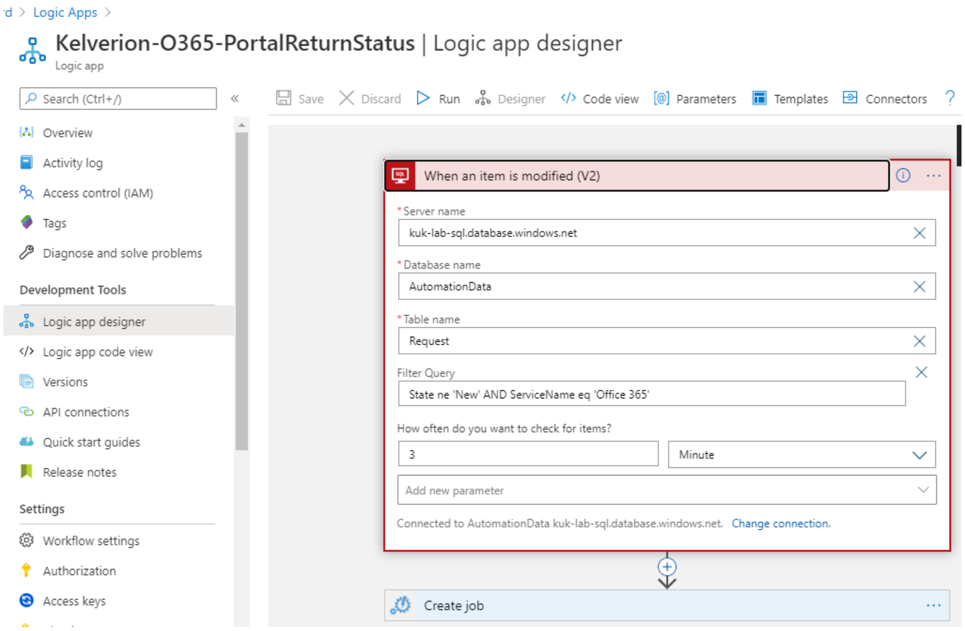


```
'Delete User' {
        $out = [PSCustomObject]@{
        Runbook = 'Kelverion_0365_10-3_Worker-Delete-User'
          'Delete User' = [PSCustomObject]@{
          UserPrincipalName = $inputConverted.'User'.'UserPrincipalName'
          ObjectId = $inputConverted.'User'.'ObjectId'
              }
          }
```

3.10.5 Returning the Results

Each application has a return runbook that gathers updates to the PDS table and returns the message and state back to the required Service Desk portal.
For the Kelverion created applications, this uses the Kelverion Automation Portal. The runbook will be launched via a Logic App in Azure that detects the changes in the dbo.Request table.





When an item is modified (V2)

* Server name: kuk-lab-sql.database.windows.net

* Database name: AutomationData

* Table name: Request

Filter Query: State ne 'New' AND ServiceName eq 'Office 365'

How often do you want to check for items?: 3 Minute

Add new parameter

Connected to AutomationData kuk-lab-sql.database.windows.net. Change connection.

N.B. The Logic App filter can be modified if more than one ServiceName exists so that a single Logic App can be used for multiple applications.

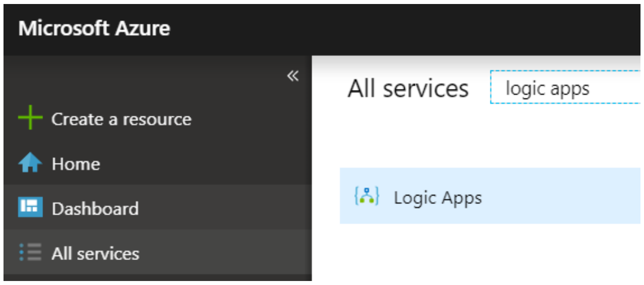
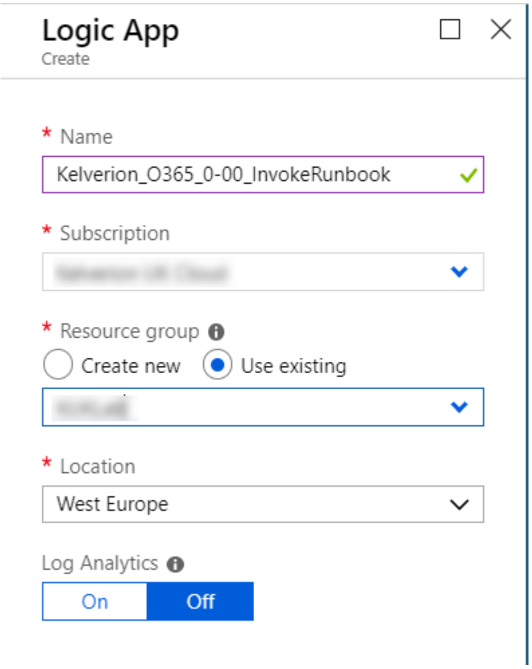
3.11 Logic Apps - Scheduling runbook execution

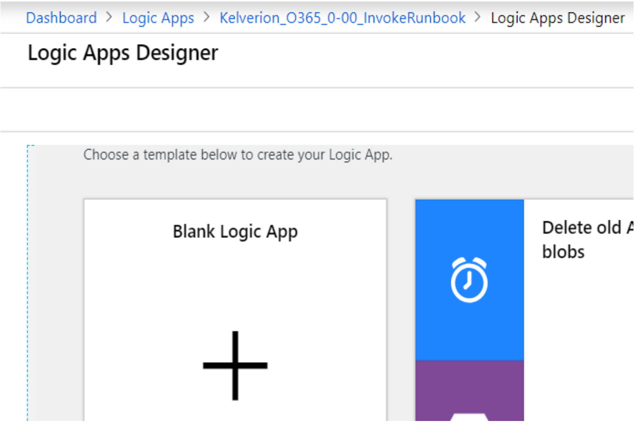
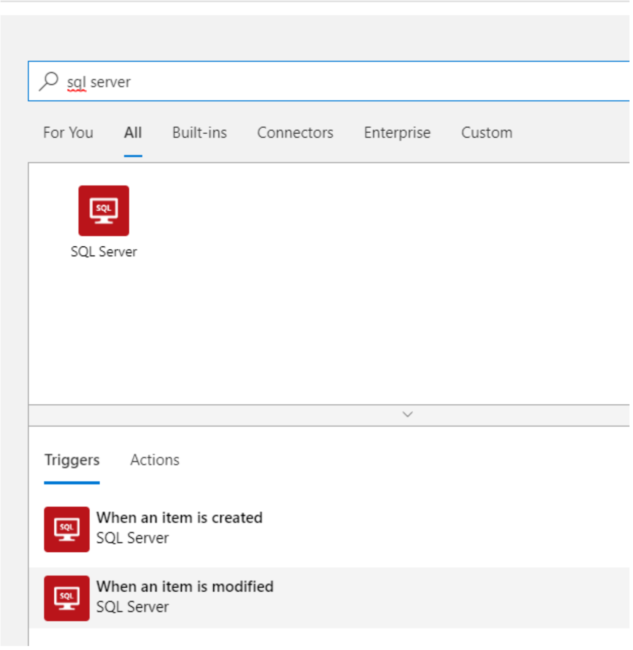
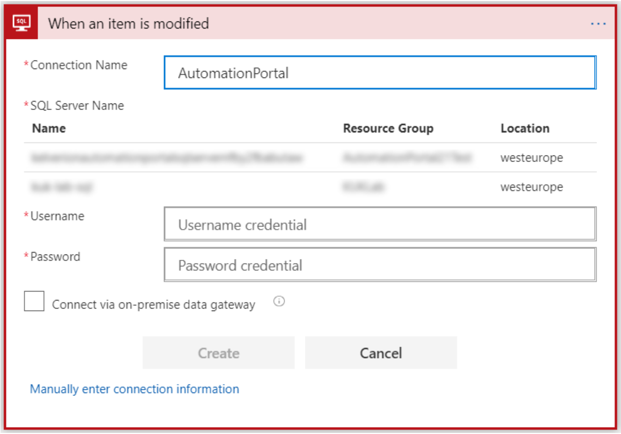
Once all the runbooks (and other assets) are in place and the runbooks have been tested you will need to schedule the runbooks for repeated execution.

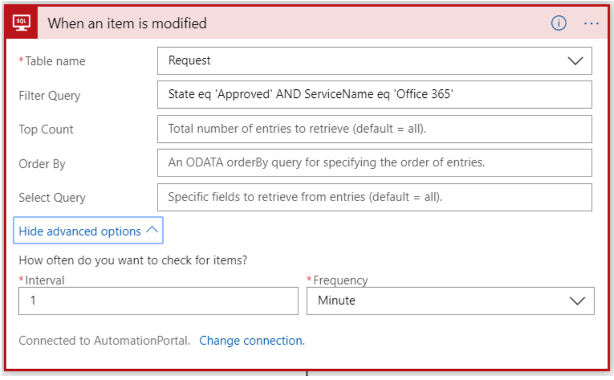
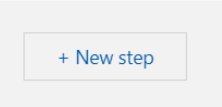
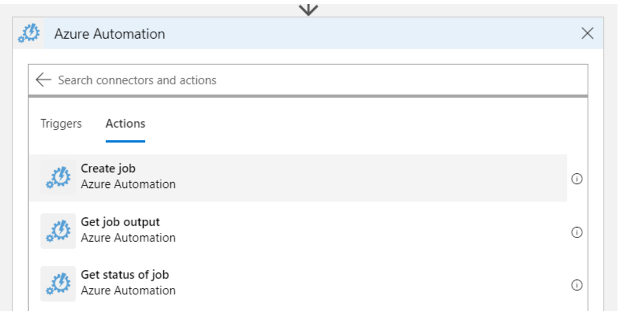
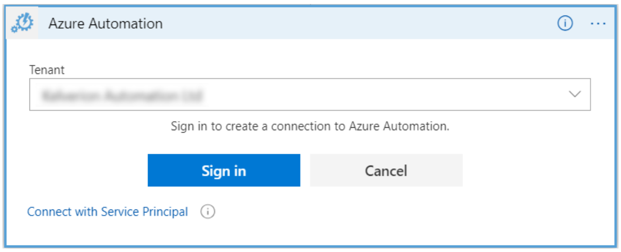
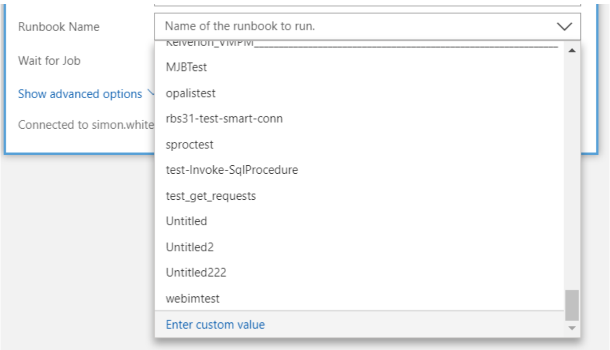
Two Logic Apps are required. One to gather the information from the required Service Desk application (in this case the Automation Portal) and one for returning the status of the runbook to the Service Desk application.

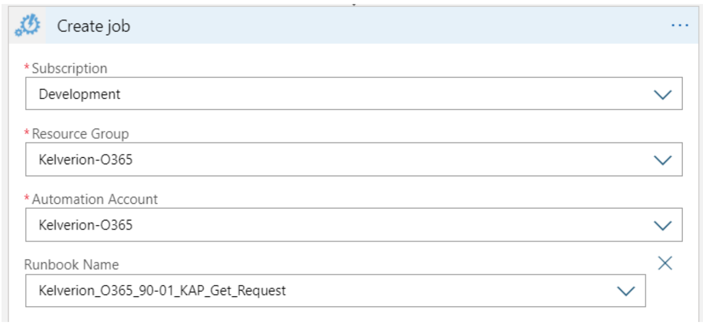
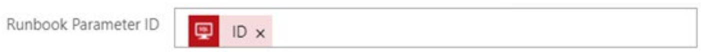
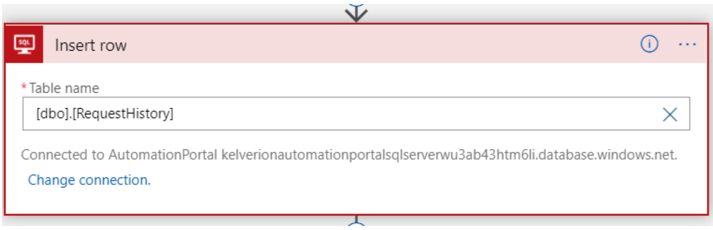
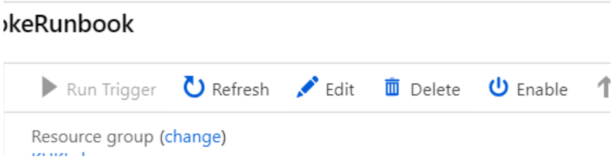
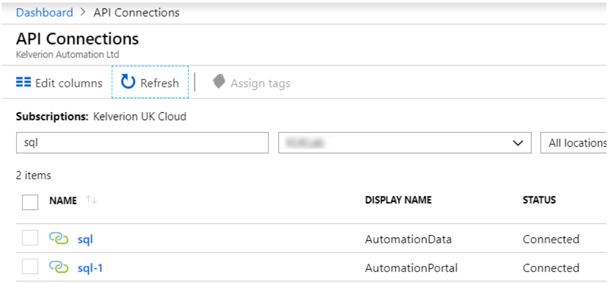
3.11.1 Gather Request Logic App

This Logic App will call the runbook Kelverion_O365_90-01_KAP_Get_Request.

Step	
Login to the Azure Portal and go to "Logic Apps"	 Microsoft Azure Create a resource Home Dashboard All services Logic Apps All services: logic apps
Create a New Logic App and name it appropriately for this application. e.g. Kelverion_O365_0-00_InvokeRunbook Add it to the same Resource Group that you have deployed the runbooks too	 * Name: Kelverion_O365_0-00_InvokeRunbook ✓ * Subscription: [redacted] * Resource group: ○ Create new ● Use existing [redacted] * Location: West Europe Log Analytics: On [✓ Off]

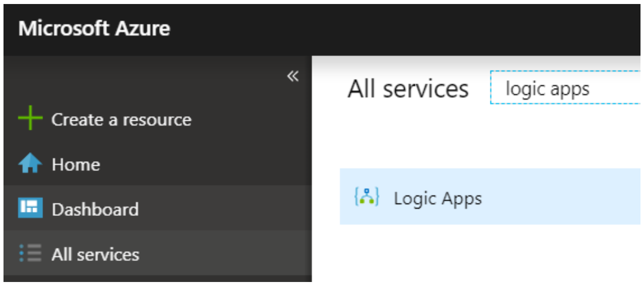
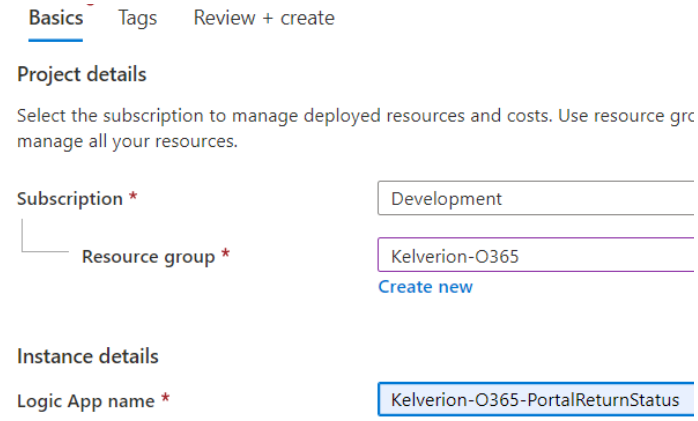
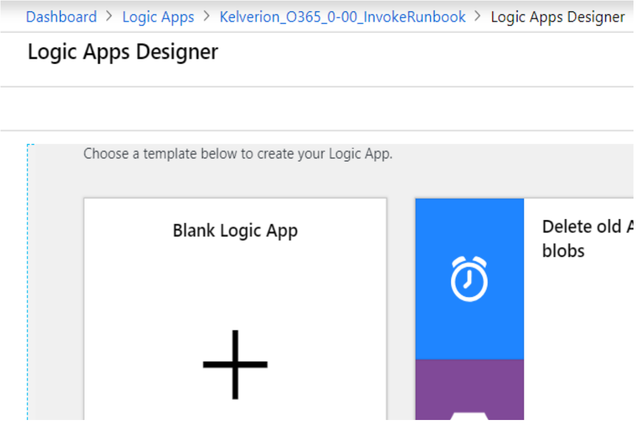
Open the newly created Logic App and select "Blank Logic App"	 Dashboard > Logic Apps > Kelverion_O365_0-00_InvokeRunbook > Logic Apps Designer Logic Apps Designer Blank Logic App Delete old A blobs Choose a template below to create your Logic App.
Search for "Sql Server" and select the Trigger "When an item is modified"	 sql server For You All Built-ins Connectors Enterprise Custom SQL Server Triggers Actions When an item is created SQL Server When an item is modified SQL Server
If you do not already have a connection (API Connection) then you will need to set one up for your Automation Portal database. Enter the appropriate connection details for the database.	 When an item is modified *Connection Name: AutomationPortal *SQL Server Name Name / Resource Group / Location westeurope westeurope *Username: Username credential *Password: Password credential ☐ Connect via on-premise data gateway

Configure the activity as shown. Table = Request Filter Query: **State eq ‘Approved’ AND ServiceName eq ‘Office 365’** Interval = 1 Minute	When an item is modified *Table name: Request Filter Query: State eq 'Approved' AND ServiceName eq 'Office 365' Top Count: Total number of entries to retrieve (default = all). Order By: An ODATA orderBy query for specifying the order of entries. Select Query: Specific fields to retrieve from entries (default = all). Hide advanced options How often do you want to check for items? *Interval: 1 *Frequency: Minute Connected to AutomationPortal. Change connection.
Click on New Step to add another activity	
Search for “Azureautomation” and select the Action “Create Job”	Azure Automation Search connectors and actions Triggers Actions Create job – Azure Automation Get job output – Azure Automation Get status of job – Azure Automation
If you have not done so before, you will need to create an API Connection to your tenant. Use the required Azure login details to make the connection.	Azure Automation Tenant: [redacted] Sign in to create a connection to Azure Automation. Sign in Cancel Connect with Service Principal
Enter the appropriate: Subscription \ Resource Group \ Automation Account For Runbook Name, scroll to the bottom and select “Custom Value”	Runbook Name: Name of the runbook to run. Wait for Job Show advanced options Connected to simon.white MJBTest opalistest rbs31-test-smart-conn sproctest test-Invoke-SqlProcedure test_get_requests Untitled Untitled2 Untitled222 webimtest Enter custom value

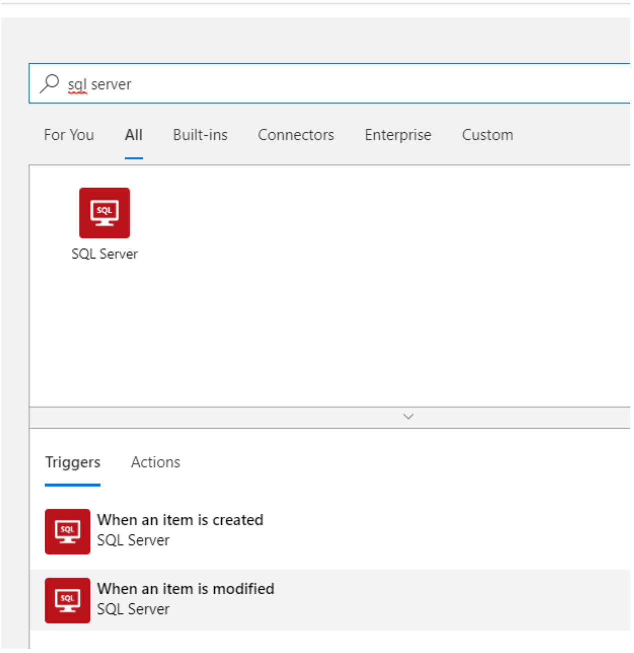
Select the Runbook from the drop down list as: **Kelverion_O365_90-01_KAP_Get_Request**	 Create job * Subscription: Development * Resource Group: Kelverion-O365 * Automation Account: Kelverion-O365 Runbook Name: Kelverion_O365_90-01_KAP_Get_Request
Add Runbook Parameters. Where ID is from the List of Items from the previous activity	 Runbook Parameter ID: ID
Automation Portal Only: Add a new activity to write a line to the RequestHistory table	 Insert row * Table name: [dbo].[RequestHistory] Connected to AutomationPortal kelverionautomationportalsqlserverwu3ab43htm6li.database.windows.net. Change connection.
Ensure the Logic App is active by clicking on **Enable**	 ›keRunbook Run Trigger Refresh Edit Delete Enable Resource group (change)
If you need to change connection details, you should be able to find your connection information in “API Connections”	 Dashboard > API Connections API Connections Kelverion Automation Ltd Edit columns · Refresh · Assign tags **Subscriptions:** Kelverion UK Cloud sql · All locations 2 items NAME · DISPLAY NAME · STATUS sql · AutomationData · Connected sql-1 · AutomationPortal · Connected

3.11.2 Return Status Logic App

This Logic App will call the runbook Kelverion_O365_97-01_KAP_Return_Status.

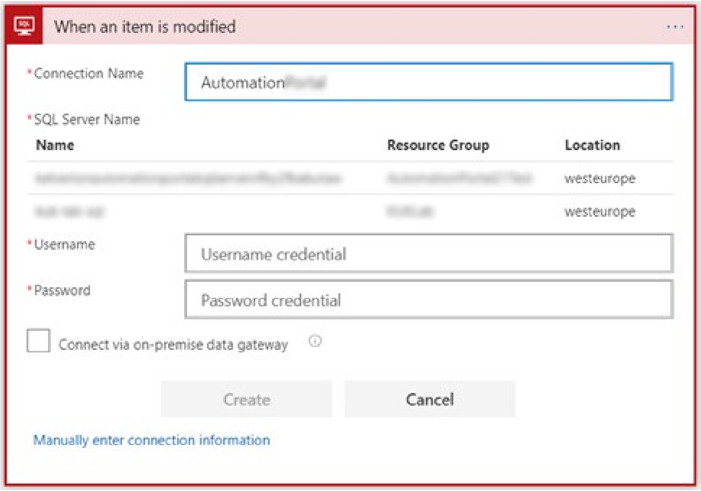
Step	
Login to the Azure Portal and go to "Logic Apps"	 Microsoft Azure « Create a resource Home Dashboard All services All services logic apps Logic Apps
Create a New Logic App and name it appropriately for this application. e.g. Kelverion-O365-PortalReurnStatus Add it to the same Resource Group that you have deployed the runbooks too	 Project details Select the subscription to manage deployed resources and costs. Use resource grc manage all your resources. Subscription * Development Resource group * Kelverion-O365 Instance details Logic App name * Kelverion-O365-PortalReturnStatus
Open the newly created Logic App and select "Blank Logic App"	 Dashboard > Logic Apps > Kelverion_O365_0-00_InvokeRunbook > Logic Apps Designer Logic Apps Designer Choose a template below to create your Logic App. Blank Logic App Delete old A blobs

Search for “Sql Server”
and
select the Trigger
“When an item is modified”



If you do not already have a connection (API Connection) then you will need to set one up for your AutomationData database.

Enter the appropriate connection details for the database.


When an item is modified

*Connection Name: Automation[redacted]

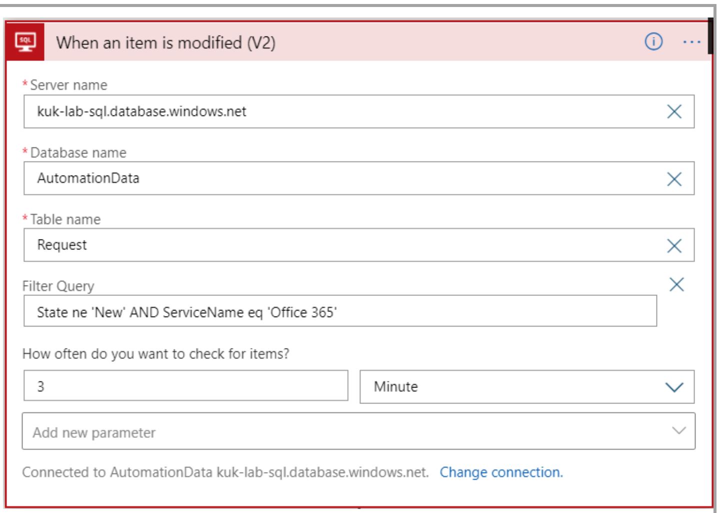
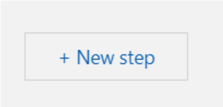
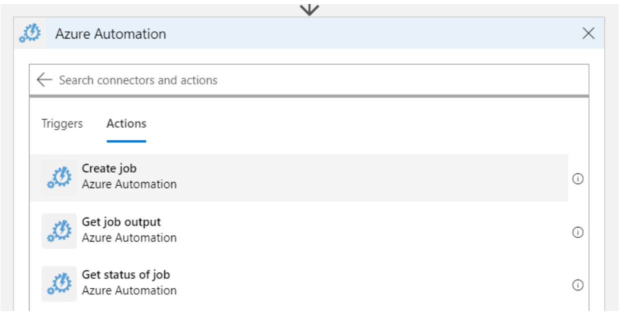
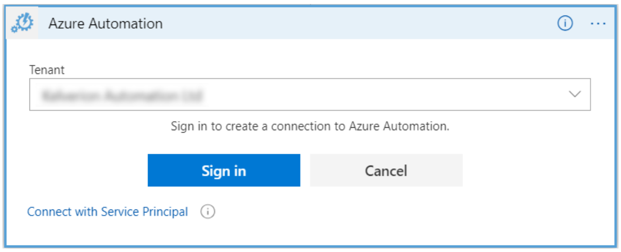
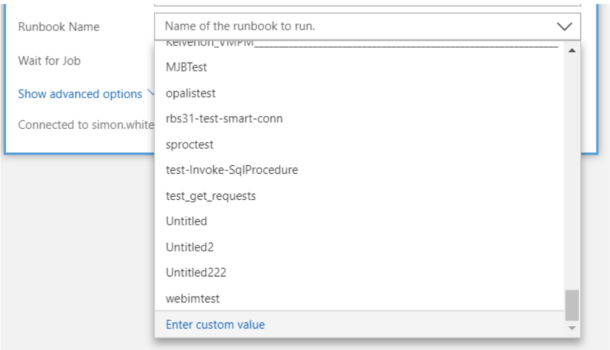
*SQL Server Name

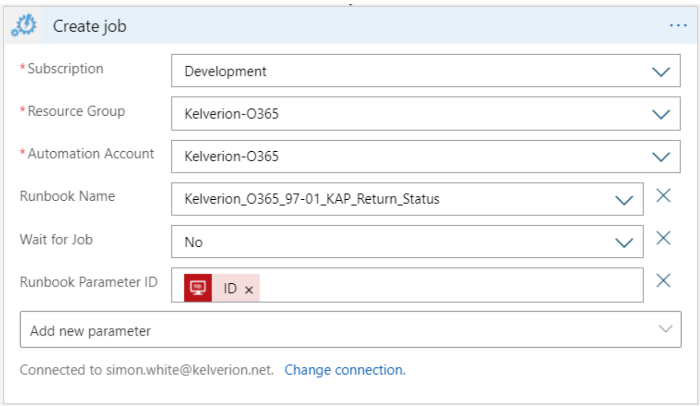
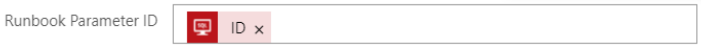
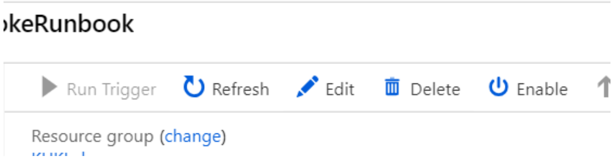
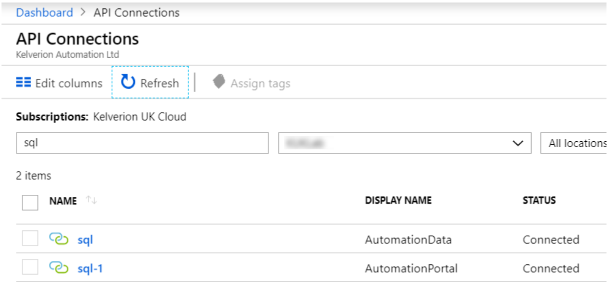
Name	Resource Group	Location
[redacted]	[redacted]	westeurope
[redacted]	[redacted]	westeurope

*Username: Username credential

*Password: Password credential

☐ Connect via on-premise data gateway

Configure the activity as shown. Table = Request Filter Query: **State ne 'New' AND ServiceName eq 'Office 365'** Interval = 1 Minute	 When an item is modified (V2) *Server name: kuk-lab-sql.database.windows.net *Database name: AutomationData *Table name: Request Filter Query: State ne 'New' AND ServiceName eq 'Office 365' How often do you want to check for items? 3 Minute Add new parameter Connected to AutomationData kuk-lab-sql.database.windows.net. Change connection.
Click on New Step to add another activity	 + New step
Search for "Azureautomation" and select the Action "Create Job"	 Azure Automation Search connectors and actions Triggers Actions Create job Azure Automation Get job output Azure Automation Get status of job Azure Automation
If you have not done so before, you will need to create an API Connection to your tenant. Use the required Azure login details to make the connection.	 Azure Automation Tenant: [redacted] Sign in to create a connection to Azure Automation. Sign in Cancel Connect with Service Principal
Enter the appropriate: Subscription \ Resource Group \ Automation Account For Runbook Name, scroll to the bottom and select "Custom Value"	 Runbook Name: Name of the runbook to run. Wait for Job Show advanced options Connected to simon.white Kelverion_VMPM MJBTest opalistest rbs31-test-smart-conn sproctest test-Invoke-SqlProcedure test_get_requests Untitled Untitled2 Untitled222 webimtest Enter custom value

Ensure that the activity is pointing at the correct Runbook: **Kelverion_0365_97-01_KAP_Return_Status**	 Create job *Subscription: Development *Resource Group: Kelverion-O365 *Automation Account: Kelverion-O365 Runbook Name: Kelverion_O365_97-01_KAP_Return_Status Wait for Job: No Runbook Parameter ID: ID Add new parameter Connected to simon.white@kelverion.net. Change connection.
Add Runbook Parameters. Where ID is from the List of Items from the previous activity.	 Runbook Parameter ID: ID
Ensure the Logic App is active by clicking on **Enable**	 keRunbook Resource group (change)
If you need to change connection details, you should be able to find your connection information in “API Connections”	 Dashboard > API Connections API Connections Kelverion Automation Ltd Subscriptions: Kelverion UK Cloud sql All locations 2 items NAME / DISPLAY NAME / STATUS sql / AutomationData / Connected sql-1 / AutomationPortal / Connected

3.12 Testing

The following steps allow you to prove that all the components have been configured correctly. Testing the components should take place before the runbooks are scheduled for repeated execution.

1. Using the Automation Portal, create a request for Create User
2. Start the runbook Kelverion_O365_10-16_Create-User using the Azure Portal
3. Monitor the runbook to ensure it completes without errors
4. Login to Office 365 Admin Center and check for the user account
5. Check the status of the request using the Automation portal
6. Attempt to log in to office 365 using the details supplied in the automation portal

Kelverion UK Limited
1 Lea Business Park,
Lower Luton Road,
Harpenden,
Hertfordshire
AL5 5EQ
Email: info@kelverion.com
Web: www.kelverion.com