Kelverion Automation
Automated Patching Application for Azure Automation

Deployment Guide

Version 1.1

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

1 Table of Contents

2 Overview

The Automated Patching application for Azure Automation, allows you to manage Azure Update Management with service offerings that significantly reduce the tasks that require use of the Azure Portal. This reduces the administrative overhead on the Azure Administrator and allows patching tasks to be delegated to other users.

The application provides a simple and flexible interface that provides various offerings. Most of the offerings would be for a Patch Administrator. They would use the application offerings to deploy clients, create patch schedules and deploy updates. Users of devices would have a single offering that would allow them to select which patching schedule they would require for their device(s).

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 Automation Patching application package contains the following elements:

- Persistent Data Store (PDS) SQL configuration script
- Automated Patching 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 following Microsoft PowerShell modules are required:

- Az
    - Az.Accounts
    - Az.Automation

  - Az.Compute
  - Az.OperationalInsights
  - Az.Resources
  - Az.SQL
  - Az.Storage

These modules are available from the PowerShell Gallery.

The following Azure features are required:

- Automation Account
  - Update Management
- Log Analytics Workspace
- SQL Server
  - SQL Server Database

3.1.3 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. Create Managed System Identity
   d. Load Integration Modules
   e. Import Runbooks
   f. Create Smart Connections
   g. Create Azure Variables
   h. Create the Log Analytics Workspace
   i. Enable the Update Management Feature
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

3.2 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_PATCH.sql*) from the package to Create the database tables and views.

3.3 Configure the Automation Portal

Import the service request definition Services_PATCH.export
The following queries must be updated to point to the customers PDS

- Patching – Schedules
- Patching – Include KB
- Patching – Exclude KB
- Patching – UM Devices
- Patching – Classifications
- Patching – Schedules Deployed

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

3.4 Create an Azure user for starting runbooks

In Azure AD create a "local" Azure AD User.
You will need to login to the portal using the user account to set the password, as the password change flag is set when the account is created.

The account should be set as an "Automation Operator" for the Automation account (Access Control (IAM) blade under the automation account). The username and password for this account will need to be configured within your Logic App so make a note of these values.

3.5 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.

See the requirements section for a list of modules.

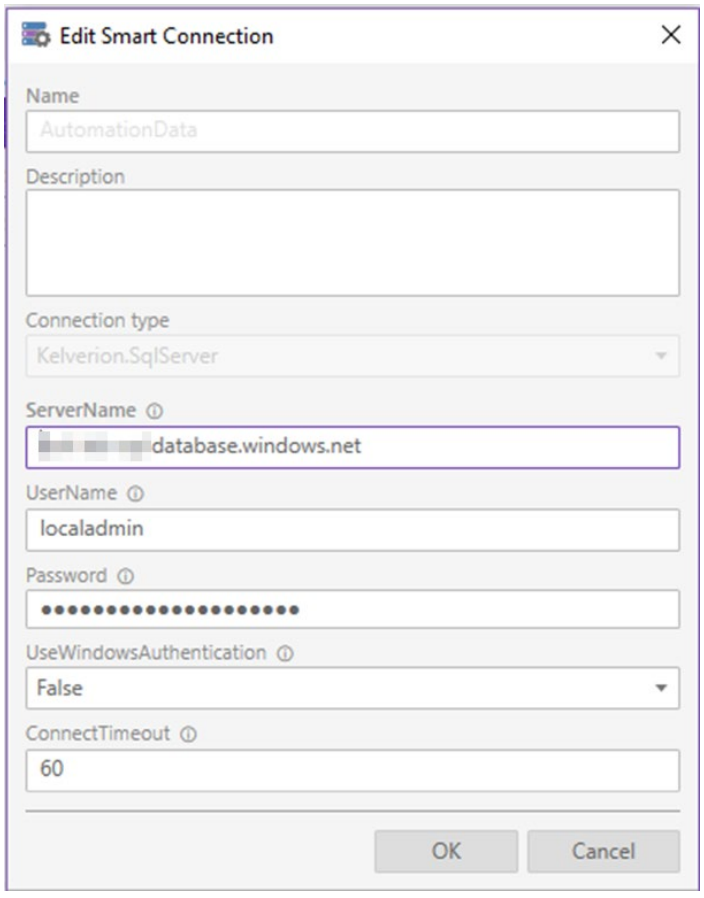
3.6 Configure your Automation account using the Runbook Studio

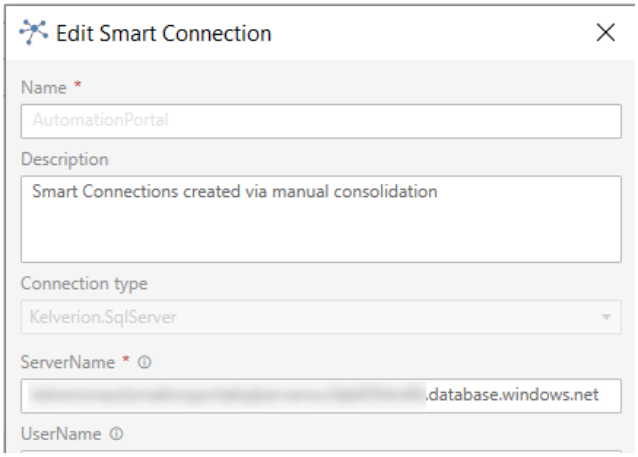
Connect the Runbook studio to your target Azure subscription. You can find more information on the initial configuration of the runbook studio on pages 5 and 6 of the User's Guide.

3.6.1 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	

AutomationPortal N.B. Only if you are using the Kelverion Automation Portal	

3.6.2 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.

Name	Description
AutomationAccount	The name of the automation account that has the runbooks \ Hybrid Worker
AzureRunAsId	This is the application ID of your Azure Run As account. N.B. If you do not have an Azure Run As account in the automation account then you will need to create one.
AzureRunAsTmbPnt	This is the thumbprint code of your Azure Run As account.
ClientId	App Id that has read writes to the Log Analytics data
ClientSecret	App Client Secret for the corresponding ClientId
LASolutionName	The name of the Update Management Log Analytics workspace
LASolutionResourceGroup	The name of the Update Management Resource Group
LASolutionSubscriptionId	The subscription of the Update Management Log Analytics workspace
LASolutionWorkspaceId	The workspace ID of the Update Management Log Analytics workspace
ResourceGroup	The name of the resource group that has the runbooks
Subscription	The subscription that the runbooks are imported into
SubscriptionID	The ID of the subscription that the runbooks are imported into
TenantId	The tenant ID of your Azure subscription that the runbook application is imported into
UpdateManagement-AutomationAccount	The automation account where the Update Management is installed
UpdateManagement-ResourceGroup	The resource group where the Update Management is installed
UpdateManagement-Subscription	The subscription where the Update Management is installed

VMSubscription	The subscription of where the client devices are. N.B. If you have devices in multiple subscriptions, then you will need to contact Kelverion for a custom application.

3.6.3 Azure Credentials

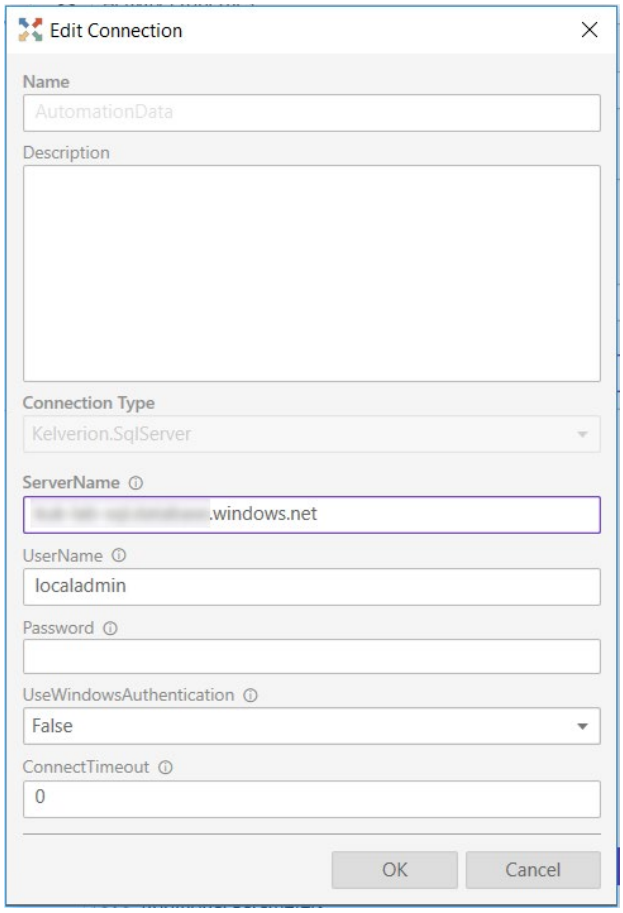
Azure Credential assets allow us to create and manage credential objects that can be utilised throughout our runbooks. These credentials are accessed by our runbooks at runtime.

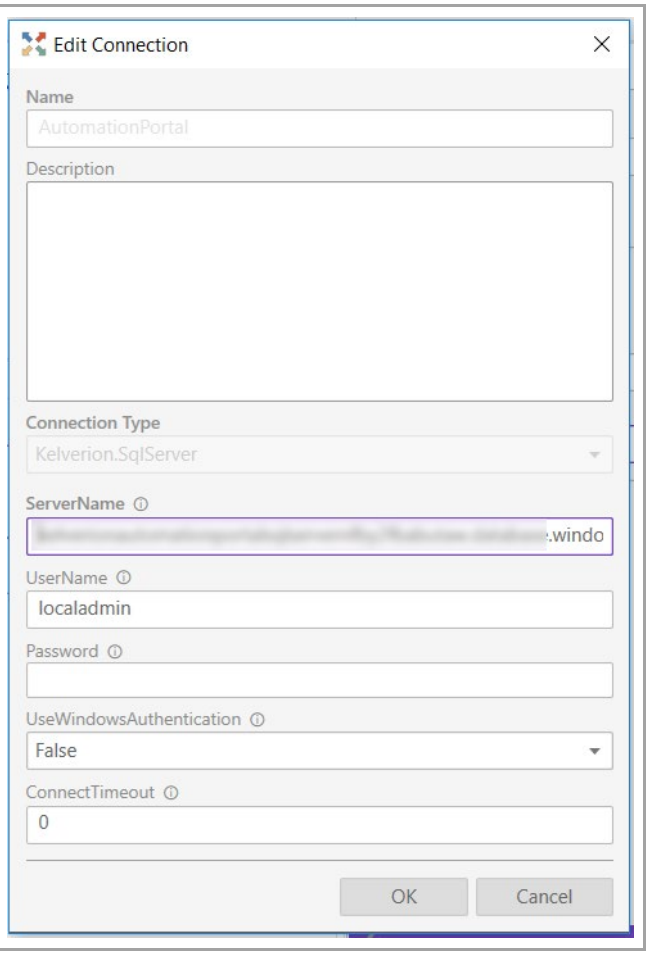
There are no Azure Credentials required for the default patching application.

3.6.4 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.	

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.	

3.7 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_PATCH_10-10_Create_PatchSchedule	This runbook will create the schedule in Azure
Kelverion_PATCH_11-10_EnableDisable_PatchSchedule	This runbook will enable \ disable a chosen schedule in Azure
Kelverion_PATCH_12-10_Create_KBList	Creates a list of KBs \ Packages used for Include \ Exclude lists in the deployment
Kelverion_PATCH_15-05_Create_PatchDeployment	Creates a patch deployment
Kelverion_PATCH_15-25_ReCreate_PatchDeployment	Runs 1hr before a deployment to update the deployment membership
Kelverion_PATCH_16-05_AddDevicesToPatchSchedule	Adds devices to the patch AD group
Kelverion_PATCH_17-05_Remove_PatchDeployment	Removes a patch deployment
Kelverion_PATCH_20-0_Util-GetRunbookData	This runbook will gather Request data from the Automation Portal
Kelverion_PATCH_30-0_Install_UMClients	Installs the Update Management client to any, powered on, device(s) in a chosen resource group
Kelverion_PATCH_80-10_UpdateScheduleTable	Discovery runbook called by a scheduled Logic App to update the schedule tables next run column
Kelverion_PATCH_80-20_Get-PatchJobs	Discovery runbook that checks the individual patch runbooks
Kelverion_PATCH_80-30_Discover_UM_Devices	Discovery runbook to keep the device list for the automation portal up to date. Called by a Logic App.
Kelverion_PATCH_90-01_KAP_Get_Request	Gathers the request data from the Kelverion Automation Portal. Requires a Logic App to trigger
Kelverion_PATCH_95-01_ConvertToRequestData	Converts the gathered data to the correct format for the runbooks

Kelverion_PATCH_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

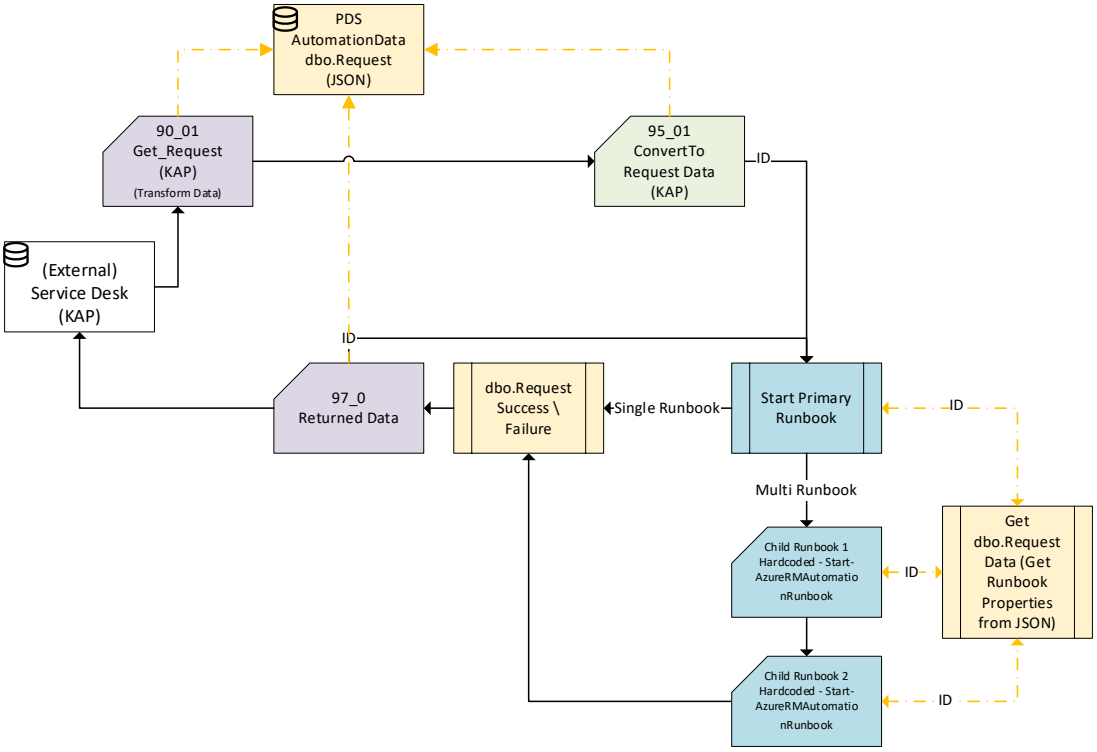
3.8 Runbook Customisation

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).

3.8.1 High Level Overview

The main process flow is as follows:



3.9 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.

3.9.1 Request Data Storage (PDS Design)

The applications use 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

3.9.1.1 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

3.9.1.2 Request Schedule Table

This table is used for scheduling any runbook deployments. The use of it is covered in the section "Scheduling Runbooks".
N.B. This table is not required for the standard automated patching application.

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

Column	Description
ID	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Standard Kelverion PDS column. E.g. Unprocessed entries are "New"
Request_ID	ID of the request offering from the dbo.Request table
Runbook	The name of the runbook that will be called
StartDate	The date \ time of when the runbook will be actioned
EndDate	The date \ time of when the runbook will be actioned
ExternalId	To store a 3rd party request reference

3.9.2 Automated Patching (PDS Design)

The following tables are used to allow the portal to provide data to the end user.

3.9.2.1 ADDevices

This table is used to store the available devices in Active Directory that can be used by the Automation Portal. N.B. This is not required for the default patching application.

Column	Description
_ID	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Standard Kelverion PDS column. E.g. Unprocessed entries are "New"
_owner	Standard Kelverion PDS column. Defines the owner or person who has updated the data
distinguishedName	Distinguished Name of the device
CompName	Name of the device
objectGUID	Device ObjectGUID
SamAccountName	Device SamAccountName
SID	Device SID

3.9.2.2 KBList

This table stores the list of KB \ Packages for inclusion \ exclusion lists.

Column	Description
_ID	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Standard Kelverion PDS column. E.g. Unprocessed entries are "New"

_owner	Standard Kelverion PDS column. Defines the owner or person who has updated the data
IncludeList	Boolean (True = Include; False = Exclude)
OS	Either Windows or Linux
ListName	Name of List
Description	List description
KBs	List of KBs \ Packages. These should be comma separated

3.9.2.3 Schedule

This table stores data about the Azure schedule and associated OMS Group

Column	Description
_ID	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Set to 'New' as default. Runbook Kelverion_PATCH_15-15_ReCreate_PatchDeployment will set this to Locked and then Active 1hr before deployment.
_owner	Standard Kelverion PDS column. Defines the owner or person who has updated the data
Enabled	(Boolean) True = Enabled and available for selection in the automation portal
Name	Name of the schedule
Description	Schedule description
RGName	Name of the resource group the Update Management automation account is in
AAName	Name of the automation account the Update Management feature is
Subscription	Name of the subscription the Update Management automation account is in
DayOfWeek	Name of the day the patch deployment occurs in the week
DayOfWeekOccurrence	(string) occurrence of the day in the month E.g. First \ Second \ Third
MonthInterval	(Int) Monthly frequency
StartTime	StartDate of the schedule and StartTime of when it will occur
ExpiryTime	EndDate Time (Optional) of the schedule
NextRun	DateTime of the next time the schedule is due to run
ADGroup	Name of the AD group associated to the schedule
Params	The JSON parameters used to create the schedule
RowVersion	SQL column for monitoring row changes
OMSGroup	Name of the associated Log Analytics saved search
OS	Targeted OS

3.9.2.4 UM Devices

This table stores data about the devices discovered in Log Analytics and provides the association for each device, to which team.

Column	Description
_ID	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp

_state	Set to 'New' as default.
_owner	Standard Kelverion PDS column. Defines the owner or person who has updated the data
Computer	The computer name. Has to be unique for Non-Azure devices.
ComputerEnvironment	Azure \ Non-Azure
SourceComputerId	Gathered from Azure Resource Discovery
VMUUID	Azure VMUUID
OSType	Discovered for Linux devices
OSName	Discovered for Linux devices
OSVersion	Discovered for Linux devices
OSFullName	Discovered for Linux devices
ResourceId	Required for Azure devices
DeviceOwner	Required to filter device per logged on user

3.9.2.5 Patch Job Log

This table stores data about the patch runbooks that are run on each device.

Column	Description
_ID	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Set to 'New' as default.
_owner	Standard Kelverion PDS column. Defines the owner or person who has updated the data
JobId	The Id of the runbook job
Job_CreationTime	The time the runbook job was created
Job_Status	The status of the runbook
Job_StartTime	The time the runbook started
Job_EndTime	The time the runbook ended
Computer	Lists the device name that the runbook ran on
OS	Lists the OS detected for the patched device
Schedule	The associated schedule that was used to patch the device
Schedule_StartTime	The associated schedules start time
Schedule_EndTime	The associated schedules end time

3.9.2.6 Portal User Team

This table stores the data for the portal users and their corresponding team. It is used to link the users to devices

Column	Description
_ID	Unique column ID. Created when data is inserted.
_created	Create time stamp
_updated	Updated time stamp
_state	Set to 'New' as default.

_owner	Standard Kelverion PDS column. Defines the owner or person who has updated the data
PortalUser	The user login ID used to connect to the Kelverion Automation Portal
TeamName	The name of the team

3.9.3 Gathering the Data

The base applications come 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.

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)

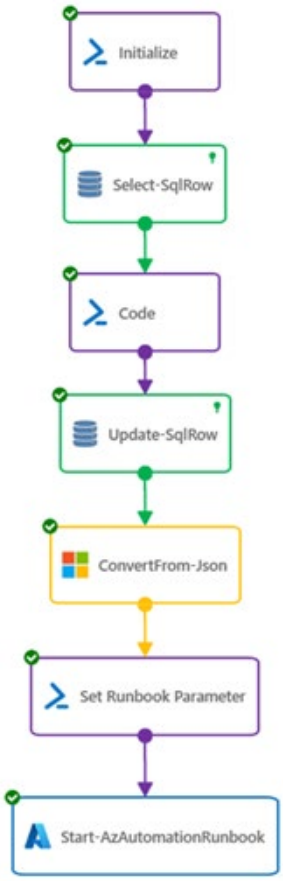
Each application has a 90-0x_XXX_Get_Request runbook that can be configured for the required Service Desk portal.

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...

3.9.4 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'
                }
            }
```

Service Desk Offering Name

Parent Runbook Name

Worker Runbook Offering Name

Worker Runbook Inputs

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.

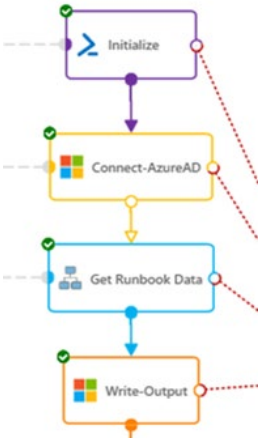
RunbookOwner	Data	ServiceName	OfferingName	ExternalId	State	OutputData
Kelverion_STSK_21-1_Work...	{ "First Name": "Bob", ...	Business User...	New Joiner	306	Complete	{ "Runbook": "Kelverion_JML_30-...
Kelverion_O365_10-4_Worke...	{ "Reason for Blocking ...	Office 365	Disable Login	305	Complete	{ "ObjectId":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

3.9.5 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:

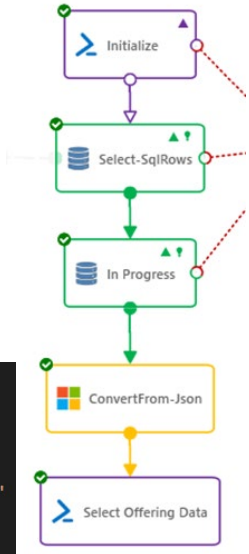
- Set some initial variables (Initialize)
- Connect to Azure (Connect-AzureAD)
- Call 'Get Runbook Data' runbook
- Output the returned data (Write-Output) for use in the main runbook activities

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.

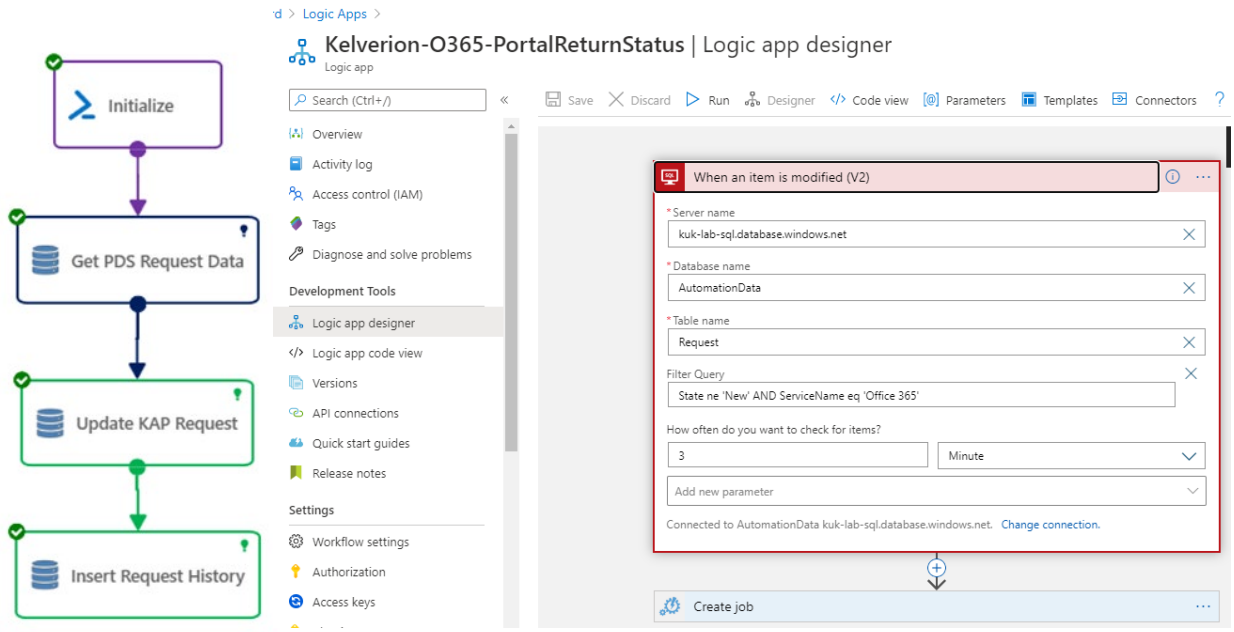



```
'Delete User' {
    $out = [PSCustomObject]@{
    Runbook = 'Kelverion_O365_10-3_Worker-Delete-User'
        'Delete User' = [PSCustomObject]@{
        UserPrincipalName = $inputConverted.'User'.'UserPrincipalName'
        ObjectId = $inputConverted.'User'.'ObjectId'
                }
          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

Service Desk Offering Name
Parent Runbook Name
Worker Runbook Offering Name
Worker Runbook Inputs

3.9.6 Returning the Results

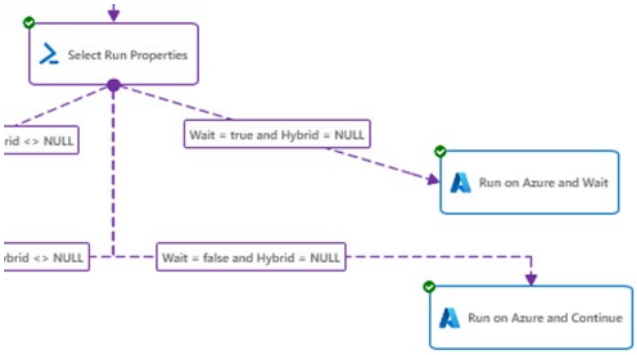
Each application has a return runbook that gathers updates to the PDS table and returns the message and state back to the required Service Desk portal.
For the Kelverion created applications, this uses the Kelverion Automation Portal. The runbook will be launched via a Logic App in Azure that detects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



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.

3.9.7 Using Hybrid Workers

For on premise runbooks you will be required to use a Hybrid Worker. Runbook Kelverion_PATCH_95-01_ConvertToRequestData has the logic already set to allow a runbook to be called using a Hybrid Worker.




```
'AD Add User to Group' {
        $out = [PSCustomObject]@{
        Runbook = 'Kelverion_STSK_21-1_Worker_AD-Add-User-To-Group'
        Hybrid = 'Hybrid-STSK'
        Wait = 'False'
            'AD Add User to Group' = [PSCustomObject]@{
            User = $inputConverted.'User'.'User'
            Group = $inputConverted.'Group'.'ADGroup'
            Domain = $inputConverted.'Domain'.'Domain'
                    }
             }
    }
```

Hybrid Worker Options

To configure this, you will need to adjust the Code activity as shown below.

If you leave the Hybrid option empty, then the runbook will assume that it is to run in Azure.

3.10 Logic Apps - Scheduling runbook execution

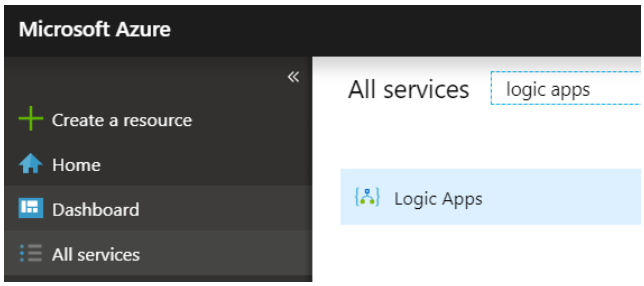
Once all the runbooks (and other assets) are in place and the runbooks have been tested you will need to schedule the 9X runbooks for repeated execution.

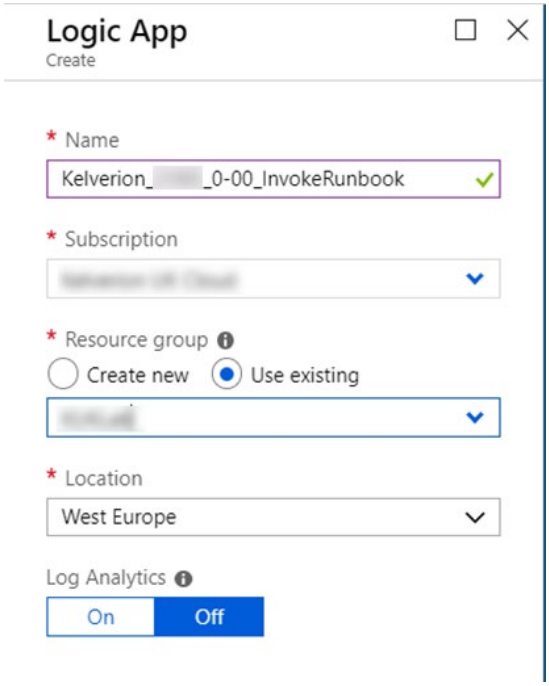
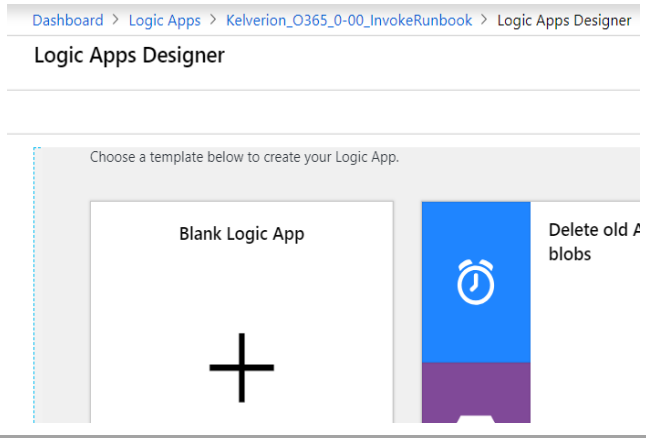
Five Logic Apps are required:

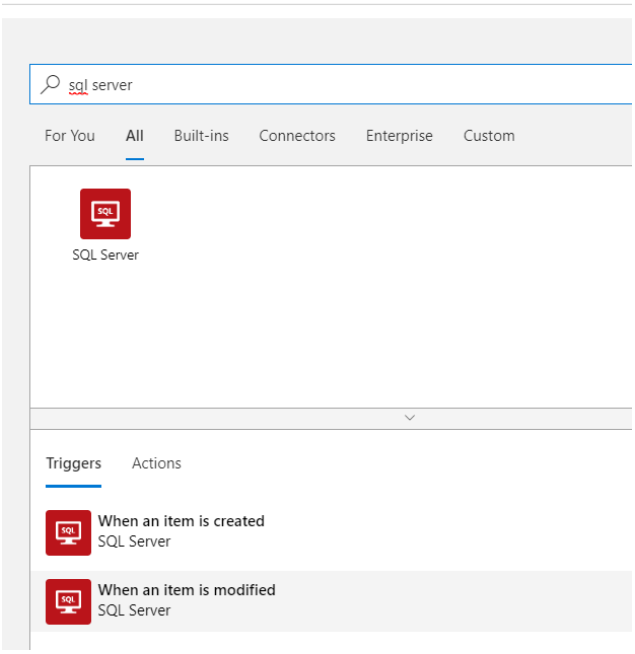
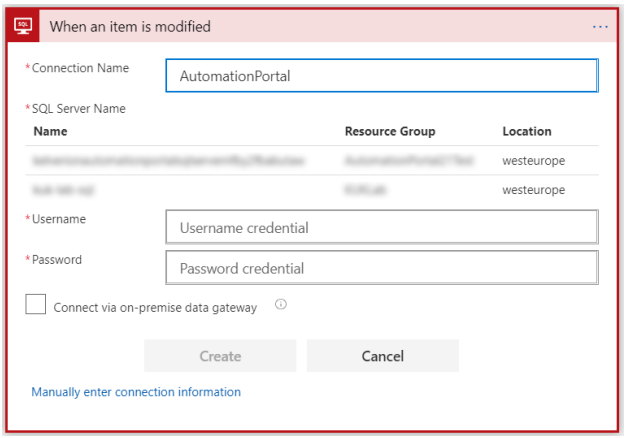
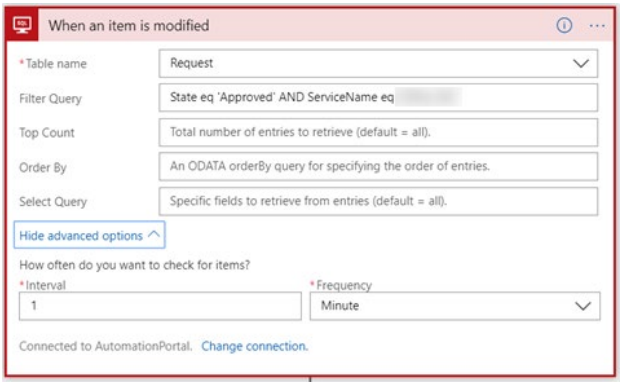
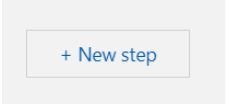
- One to gather the information from the required Service Desk application (in this case the Automation Portal). Kelverion_PATCH_90-01
- One for returning the status of the runbook to the Service Desk application. Kelverion_PATCH_97-01
- One for running daily discovery runbooks:
  - Kelverion_PATCH_80-10
  - Kelverion_PATCH_80-30
- One for hourly patch discovery runbook:
  - Kelverion_PATCH_80-20
- One for checking the SQL view v_NextRun
  - Schedule every 5 mins
  - Get rows (V2)

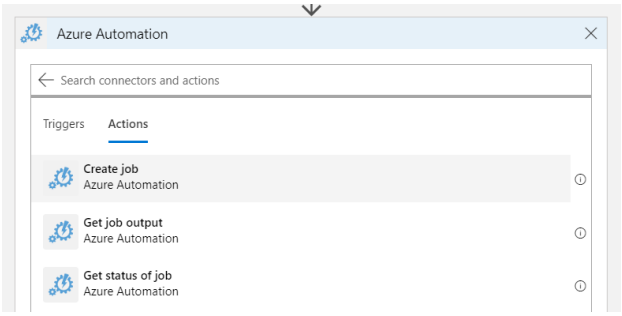
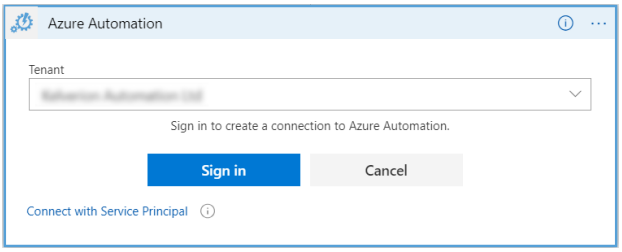
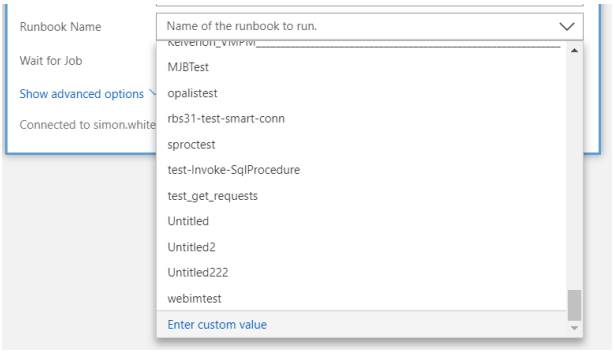
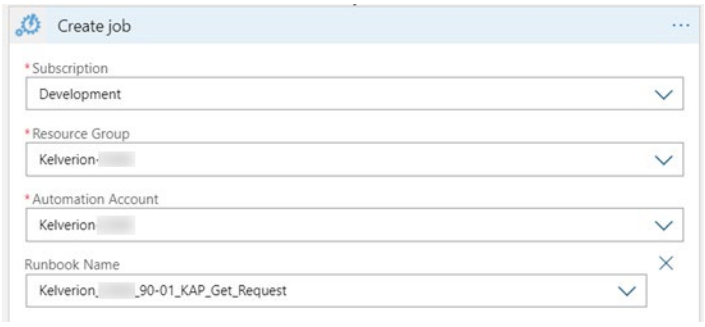
3.10.1 Gather Request Logic App

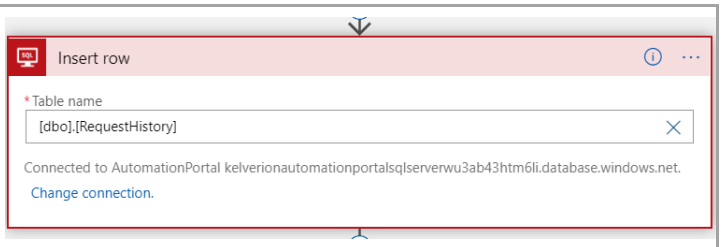
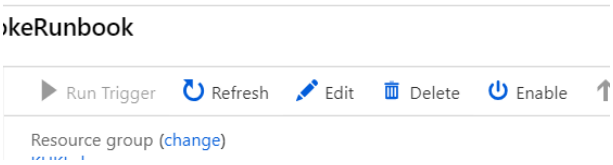
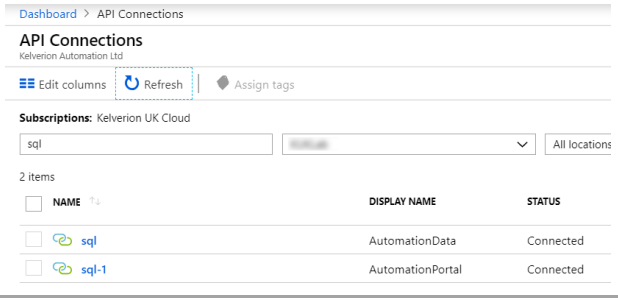
This Logic App will call the runbook Kelverion_PATCH_90-01_KAP_Get_Request.

Step	
Login to the Azure Portal and go to “Logic Apps”	

Create a New Logic App and name it appropriately for this application. e.g. Kelverion_PATCH_0-00_InvokeRunbook Add it to the same Resource Group that you have deployed the runbooks too	
Open the newly created Logic App and select "Blank Logic App"	

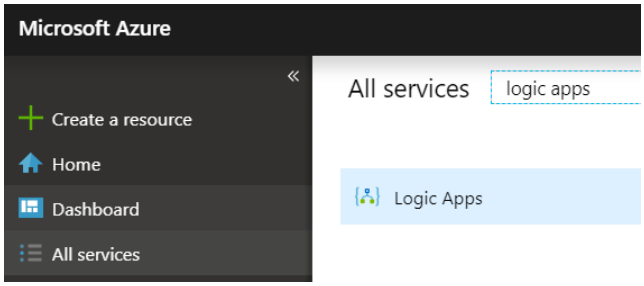
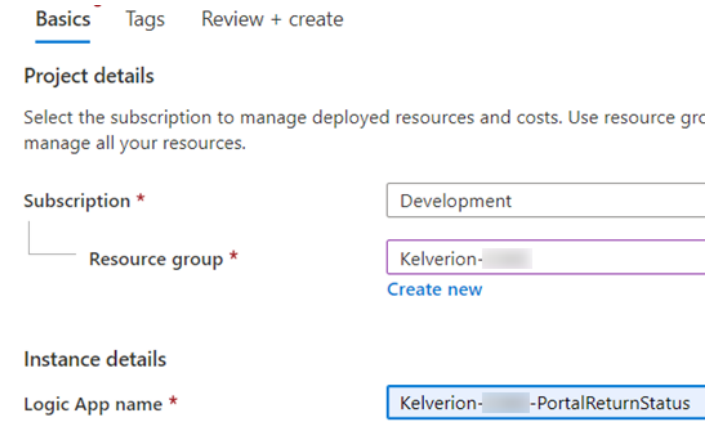
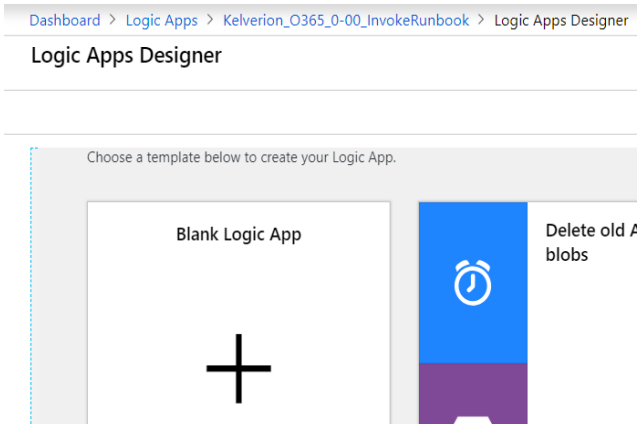
Search for “Sql Server” and select the Trigger “When an item is modified”	
If you do not already have a connection (API Connection) then you will need to set one up for your Automation Portal database. Enter the appropriate connection details for the database.	
Configure the activity as shown. Table = Request Filter Query: **State eq ‘Approved’ AND ServiceName eq ‘Automated Patching’** Interval = 1 Minute	
Click on New Step to add another activity	

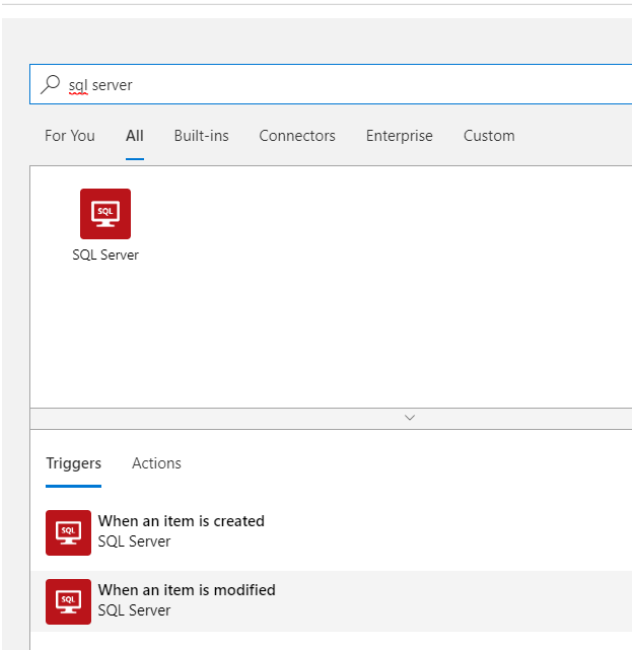
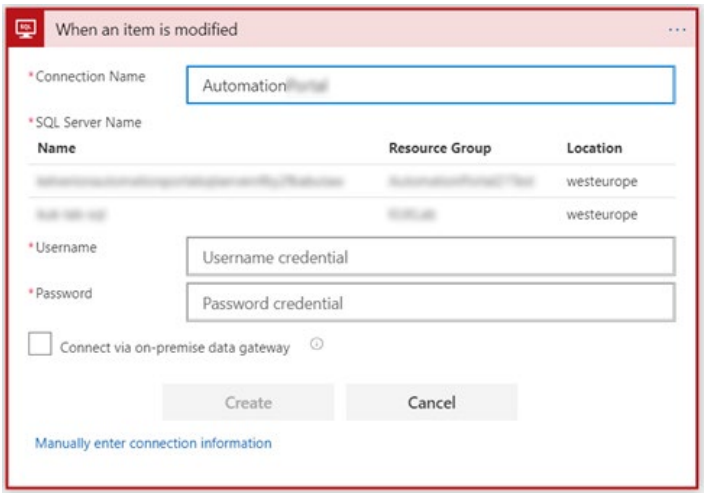
Search for “Azure Automation” and select the Action “Create Job”	
If you have not done so before, you will need to create an API Connection to your tenant. Use the required Azure login details to make the connection.	
Enter the appropriate: Subscription \ Resource Group \ Automation Account For Runbook Name, scroll to the bottom and select “Custom Value”	
Select the Runbook from the drop down list as: **Kelverion_PATCH_90-01_KAP_Get_Request**	
Add Runbook Parameters. Where ID is from the List of Items from the previous activity	

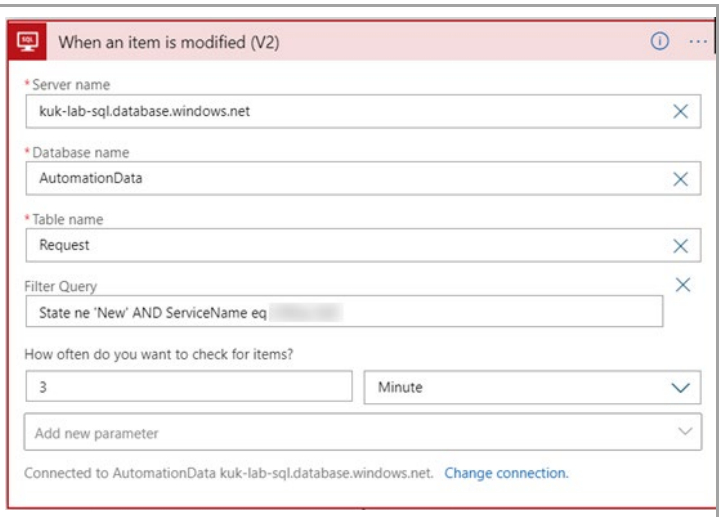
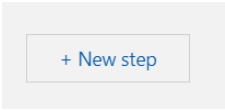
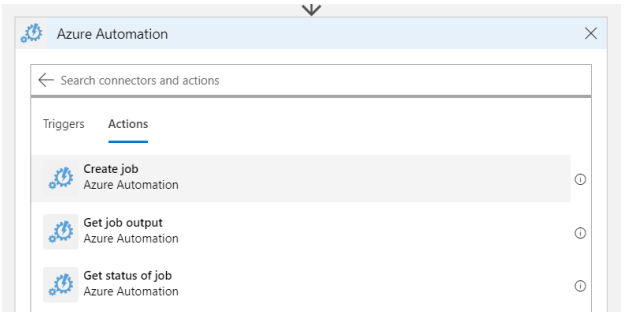
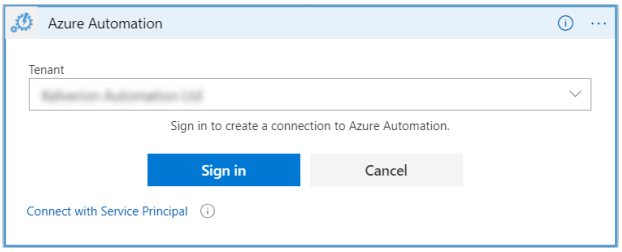
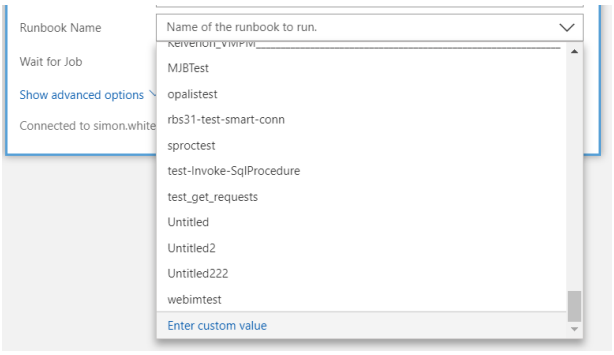
Automation Portal Only: Add a new activity to write a line to the RequestHistory table	
Ensure the Logic App is active by clicking on **Enable**	
If you need to change connection details, you should be able to find your connection information in "API Connections"	

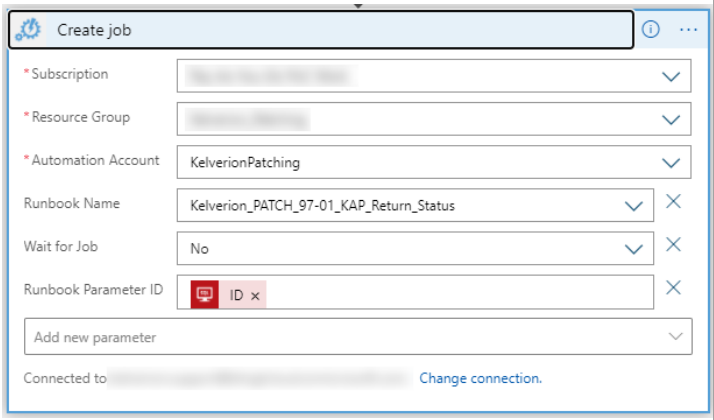
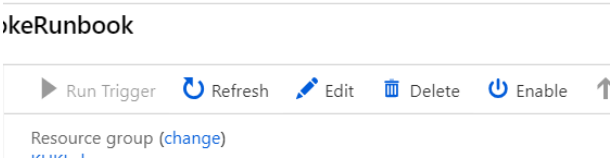
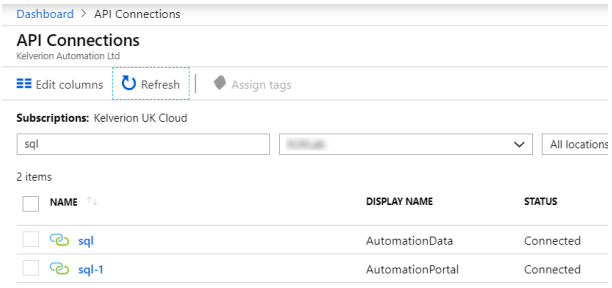
3.10.2 Return Status Logic App

This Logic App will call the runbook Kelverion_PATCH_97-01_KAP_Return_Status.

Step	
Login to the Azure Portal and go to “Logic Apps”	
Create a New Logic App and name it appropriately for this application. e.g. Kelverion-PATCH-PortalReturnStatus Add it to the same Resource Group that you have deployed the runbooks too	
Open the newly created Logic App and select “Blank Logic App”	

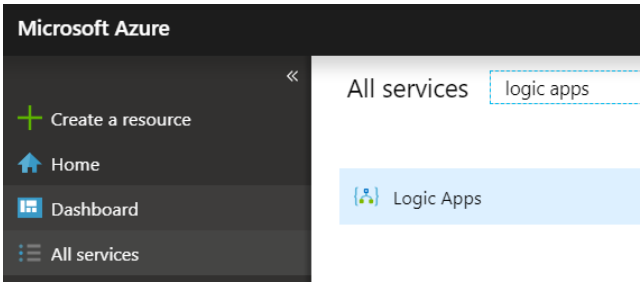
Search for “Sql Server” and select the Trigger “When an item is modified”	
If you do not already have a connection (API Connection) then you will need to set one up for your AutomationData database. Enter the appropriate connection details for the database.	

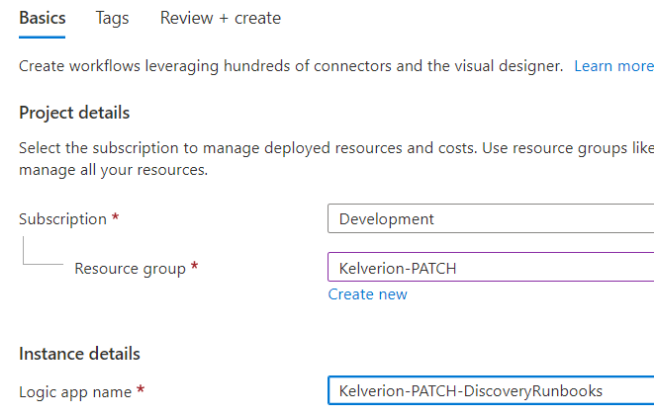
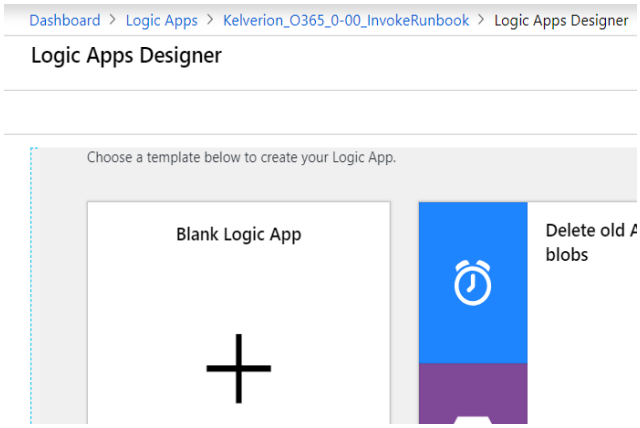
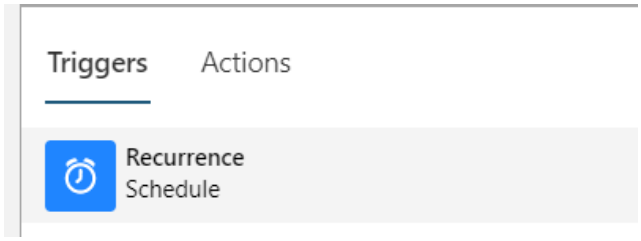
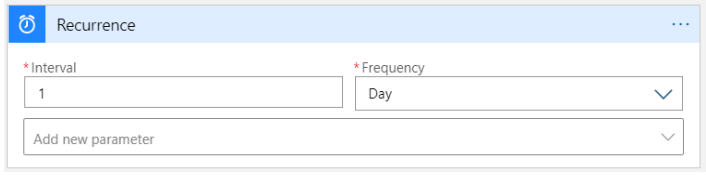
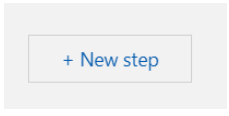
Configure the activity as shown. Table = Request Filter Query: **State ne 'New' AND ServiceName eq 'Automated Patching'** Interval = 1 Minute	
Click on New Step to add another activity	
Search for "Azure Automation" and select the Action "Create Job"	
If you have not done so before, you will need to create an API Connection to your tenant. Use the required Azure login details to make the connection.	
Enter the appropriate: Subscription \ Resource Group \ Automation Account For Runbook Name, scroll to the bottom and select "Custom Value"	

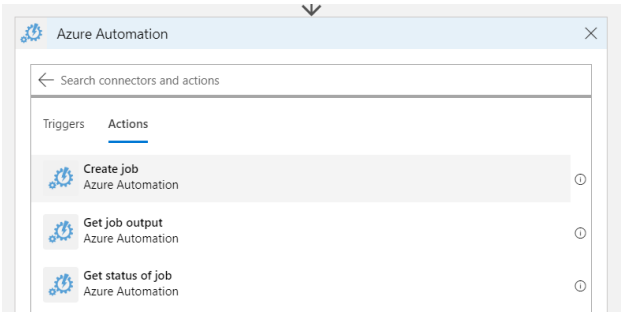
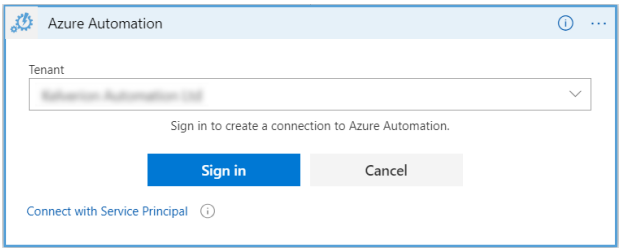
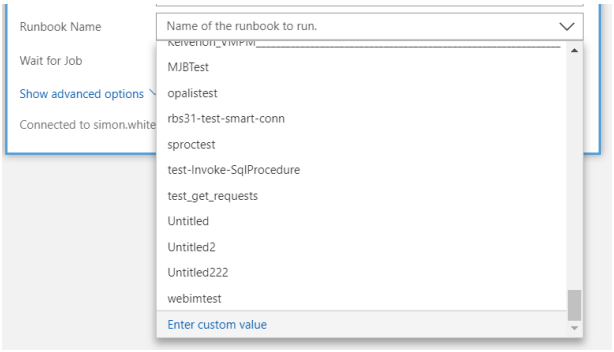
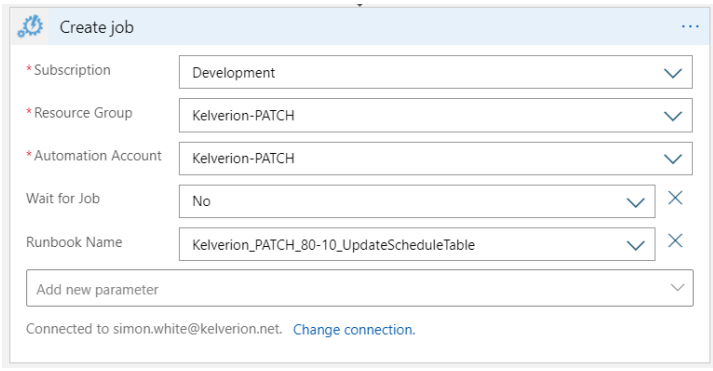
Ensure that the activity is pointing at the correct Runbook: **Kelverion_PATCH_97-01_KAP_Return_Status**	
Add Runbook Parameters. Where ID is from the List of Items from the previous activity.	
Ensure the Logic App is active by clicking on **Enable**	
If you need to change connection details, you should be able to find your connection information in "API Connections"	

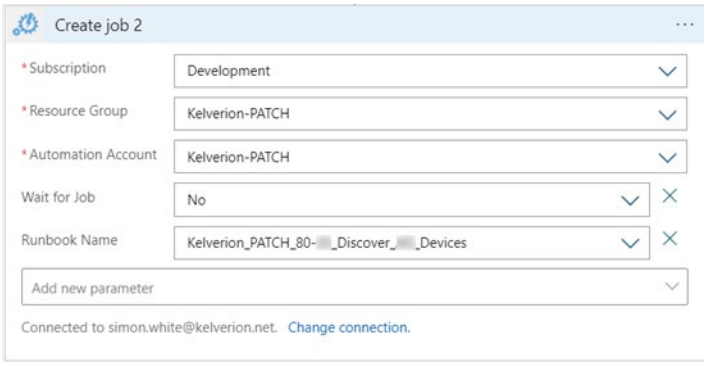
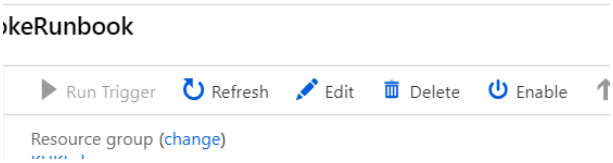
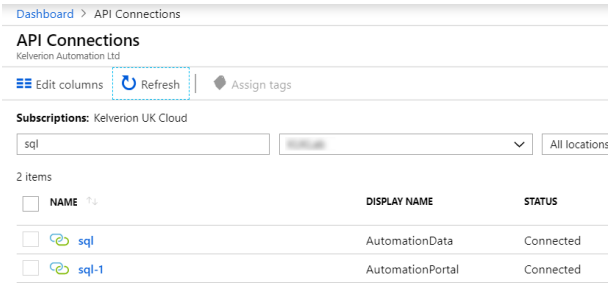
3.10.3 Discovery Logic App

This Logic App will call the discovery runbooks on a schedule.

Step	
Login to the Azure Portal and go to "Logic Apps"	

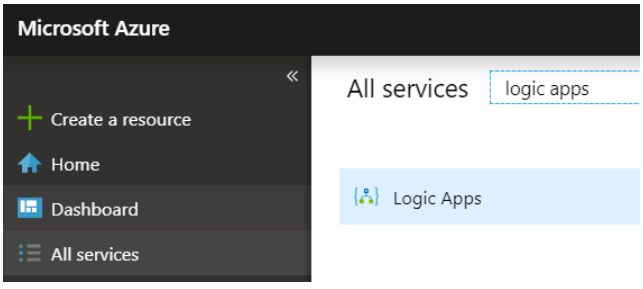
Create a New Logic App and name it appropriately for this application. e.g. Kelverion-PATCH-DiscoveryRunbooks Add it to the same Resource Group that you have deployed the runbooks too	
Open the newly created Logic App and select “Blank Logic App”	
Search for “Schedule” and select the Trigger “Recurrence”	
Set the chosen interval. Daily should be enough to cover most device discovery. You may need a more frequent discovery for your environment.	
Click on New Step to add another activity	

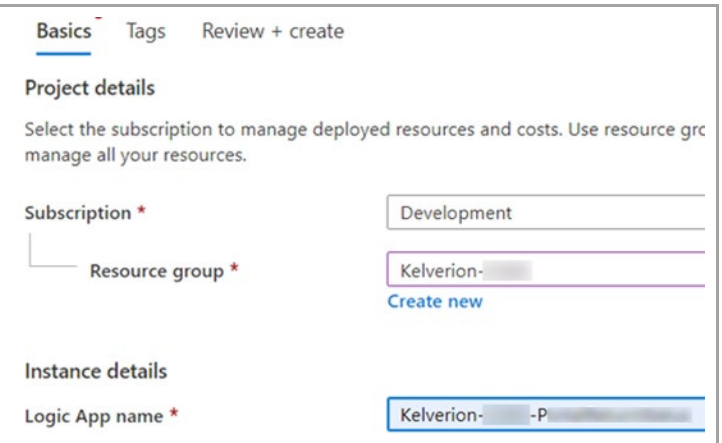
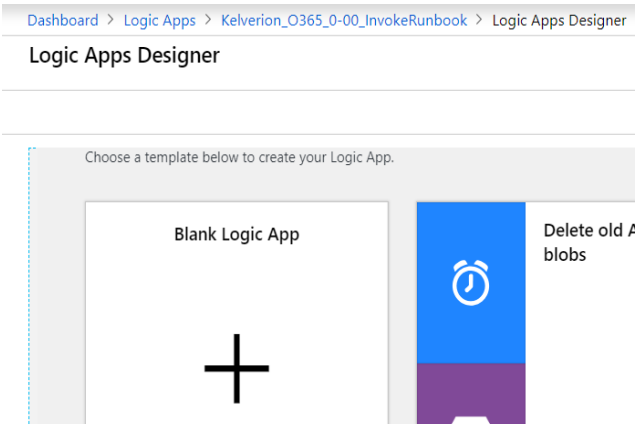
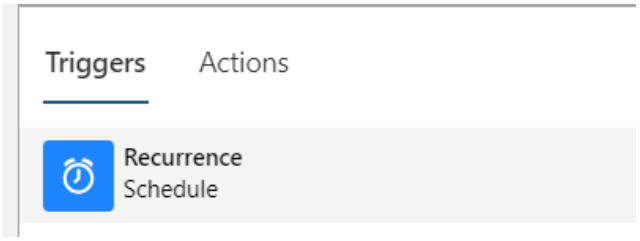
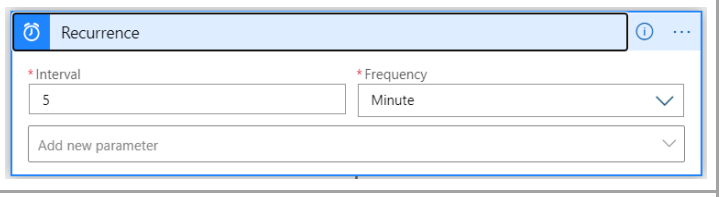
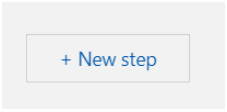
Search for “Azure Automation” and select the Action “Create Job”	
If you have not done so before, you will need to create an API Connection to your tenant. Use the required Azure login details to make the connection.	
Enter the appropriate: Subscription \ Resource Group \ Automation Account For Runbook Name, scroll to the bottom and select “Custom Value”	
Ensure that the activity is pointing at the correct Runbook: **Kelverion_PATCH_80-10_UpdateScheduleTable**	

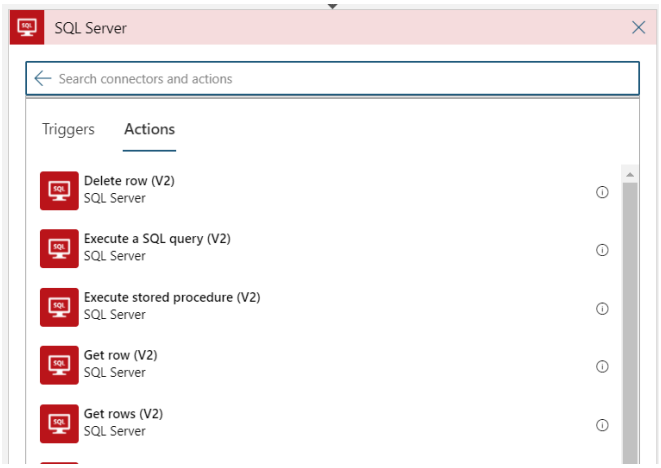
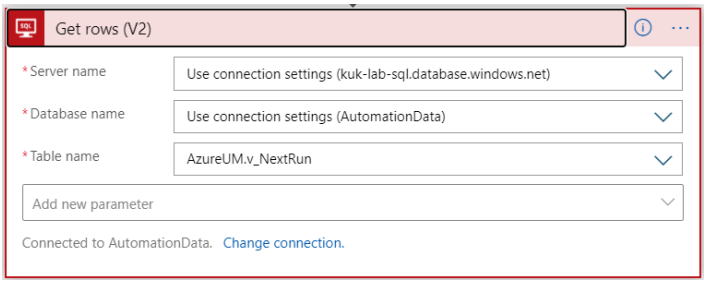
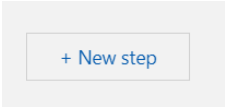
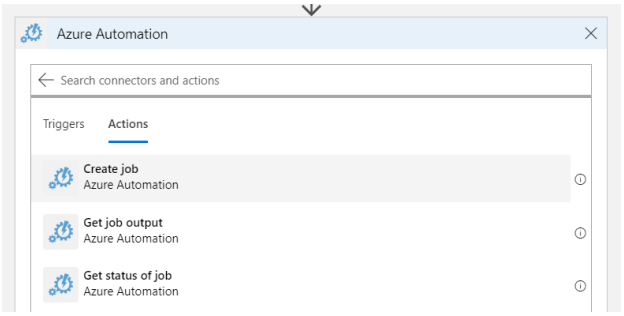
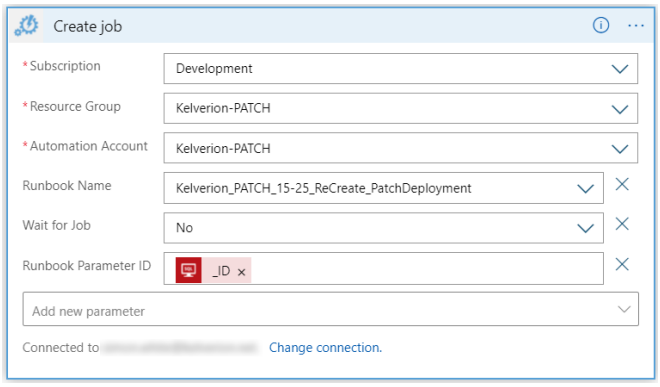
Repeat the above step for the 2nd Discovery runbook **Kelverion_PATCH_80-30_Discover_UM_Devices**	
Ensure the Logic App is active by clicking on **Enable**	
If you need to change connection details, you should be able to find your connection information in "API Connections"	

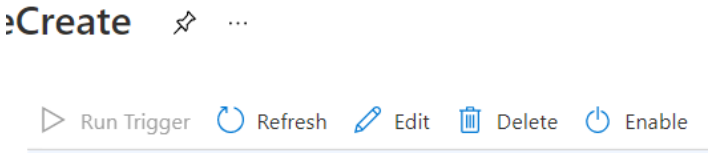
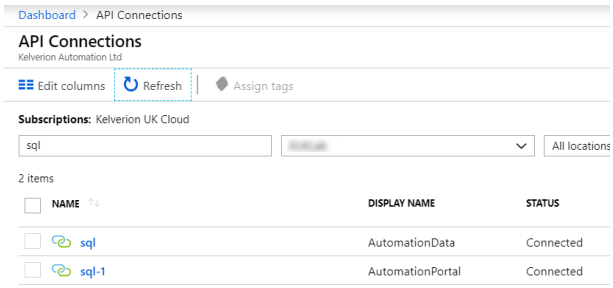
3.10.4 Patch Recreate Logic App

This Logic App will check the SQL view on a schedule.

Step	
Login to the Azure Portal and go to "Logic Apps"	

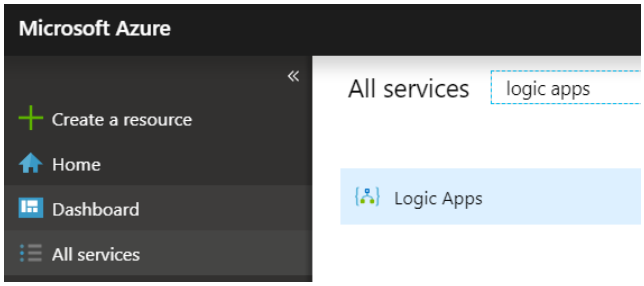
Create a New Logic App and name it appropriately for this application. e.g. **Kelverion-PATCH-Recreate** Add it to the same Resource Group that you have deployed the runbooks too	
Open the newly created Logic App and select “Blank Logic App”	
Search for “Schedule” and select the Trigger “Recurrence”	
Set the chosen interval to 5 minutes	
Click on New Step to add another activity	

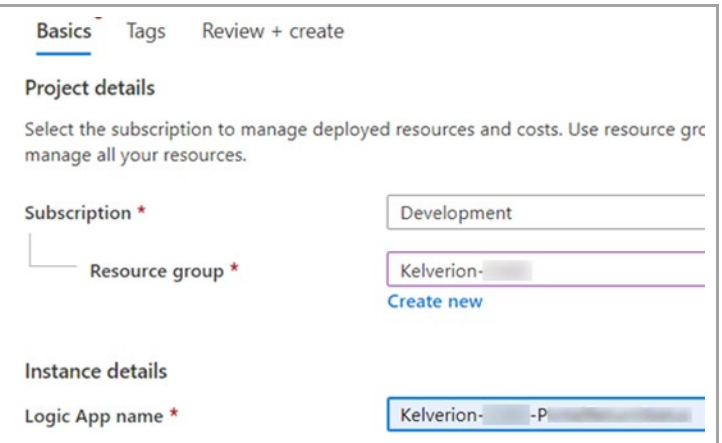
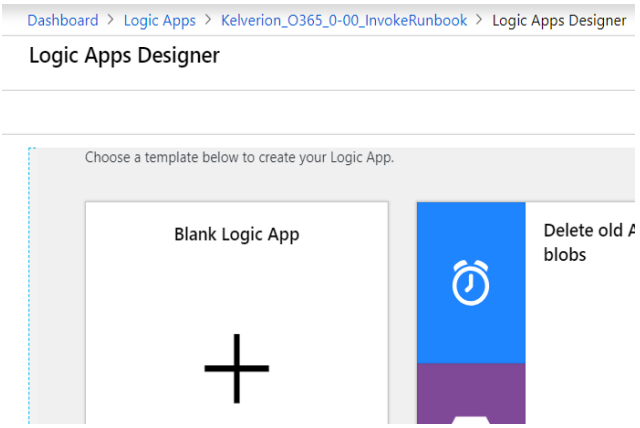
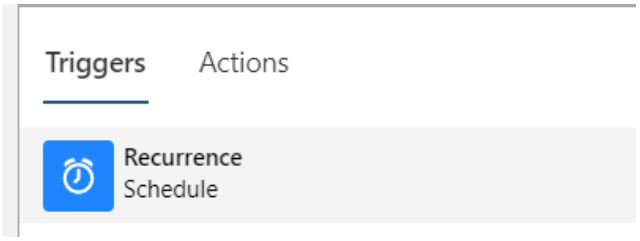
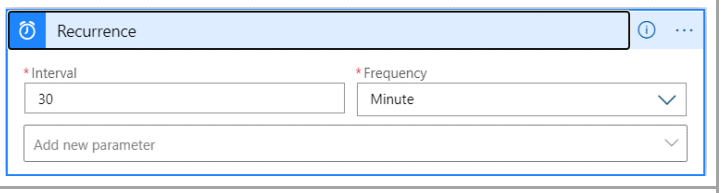
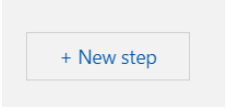
Search for “SQL” and select the Action “Get Rows (V2)”	
Configure the Get rows (V2) to look at the table name **AzureUM.v_NextRun**	
Click on New Step to add another activity	
Search for “Azureautomation” and select the Action “Create Job”	
Configure the activity to call the runbook **Kelverion_PATCH_15-25_ReCreate_PatchDeployment**	

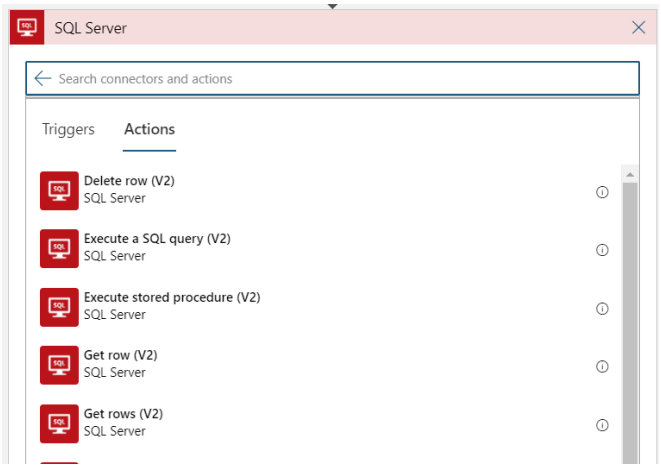
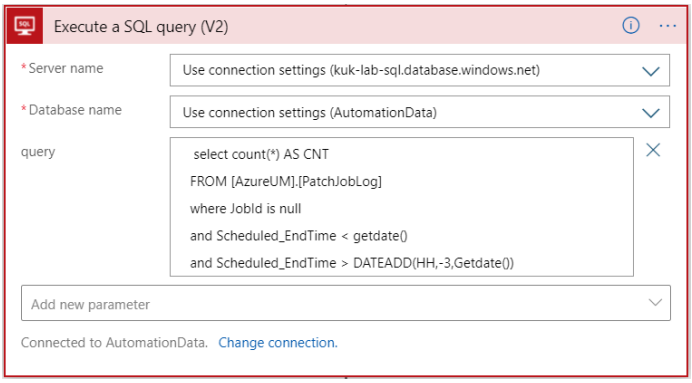
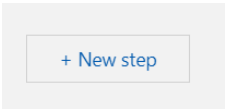
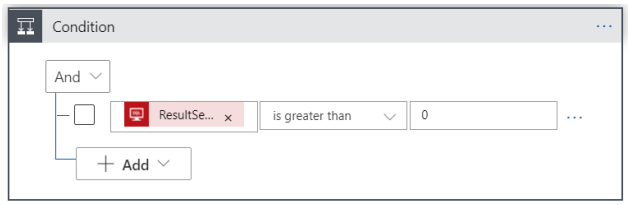
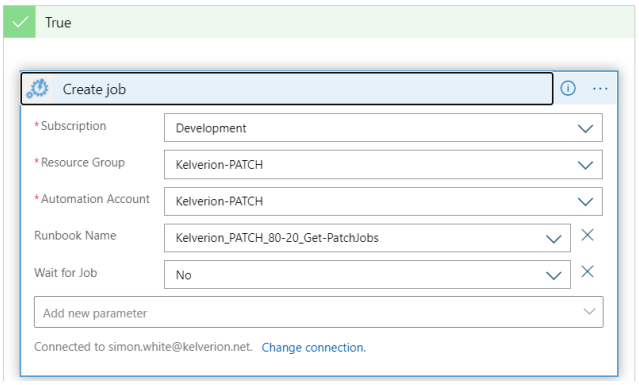
Add Runbook Parameters. Where ID is from the List of Items from the previous activity. N.B. This will add a foreach loop automatically into the logic app.	
Ensure the Logic App is active by clicking on **Enable**	
If you need to change connection details, you should be able to find your connection information in "API Connections"	

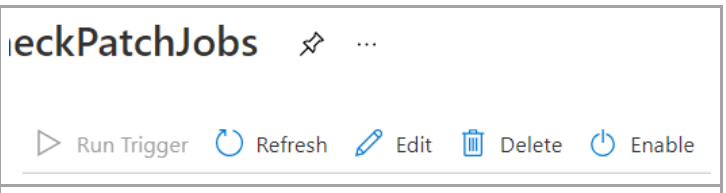
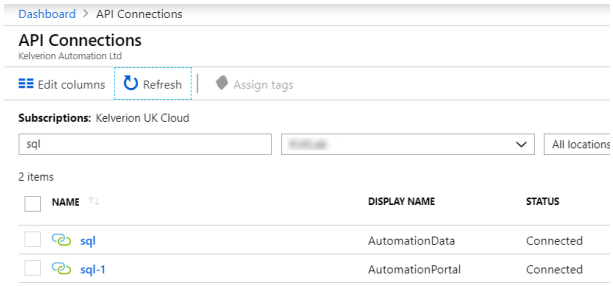
3.10.5 Patch Check Job Logs Logic App

This Logic App will check a SQL query and then call a runbook if required.

Step	
Login to the Azure Portal and go to "Logic Apps"	

Create a New Logic App and name it appropriately for this application. e.g. **Kelverion-PATCH-CheckPatchJobs** Add it to the same Resource Group that you have deployed the runbooks too	
Open the newly created Logic App and select “Blank Logic App”	
Search for “Schedule” and select the Trigger “Recurrence”	
Set the chosen interval to 30 minutes or 1 hour depending on how frequent you want patch jobs checked.	
Click on New Step to add another activity	

Search for “SQL” and select the Action “Execute a SQL query (V2)”	
Configure the query to have: select count(*) AS CNT FROM [AzureUM].[PatchJobLog] where JobId is null and Scheduled_EndTime < getdate() and Scheduled_EndTime > DATEADD(HH,-3,Getdate())	
Click on New Step to add another activity	
Add a Conditional activity and set the output results to be greater than 0.	
Configure the **True** branch to add an automation job. Set the runbook to be: **Kelverion_PATCH_80-20_Get-PatchJobs**	

Ensure the Logic App is active by clicking on **Enable**	
If you need to change connection details, you should be able to find your connection information in "API Connections"	

3.11 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 Patching Schedule
2. Start the runbook Kelverion_PATCH_90-01_KAP_Get_Request using the Azure Portal
3. Monitor the runbook to ensure it completes without errors
4. Check the resource group has created the schedule
5. Check the status of the request using 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

