Using a Persistent Data Store with Azure Automation

Best Practices Guide

Kelverion

Table of Contents

Purpose

It is Kelverion's recommendation that in nearly every case, Azure Automation runbooks should be implemented utilizing a small Azure SQL database. This is what we refer to as a Persistent Data Store (a.k.a. PDS).

There are multiple reasons why this is a valuable practice, and we will go into detail on each one:

- Persistence
- Passing Data Efficiently Between Runbooks
- Manageability
- Re-Running and Troubleshooting Runbooks
- Waiting and Manual Intervention
- Error Handling
- Speed of New Development

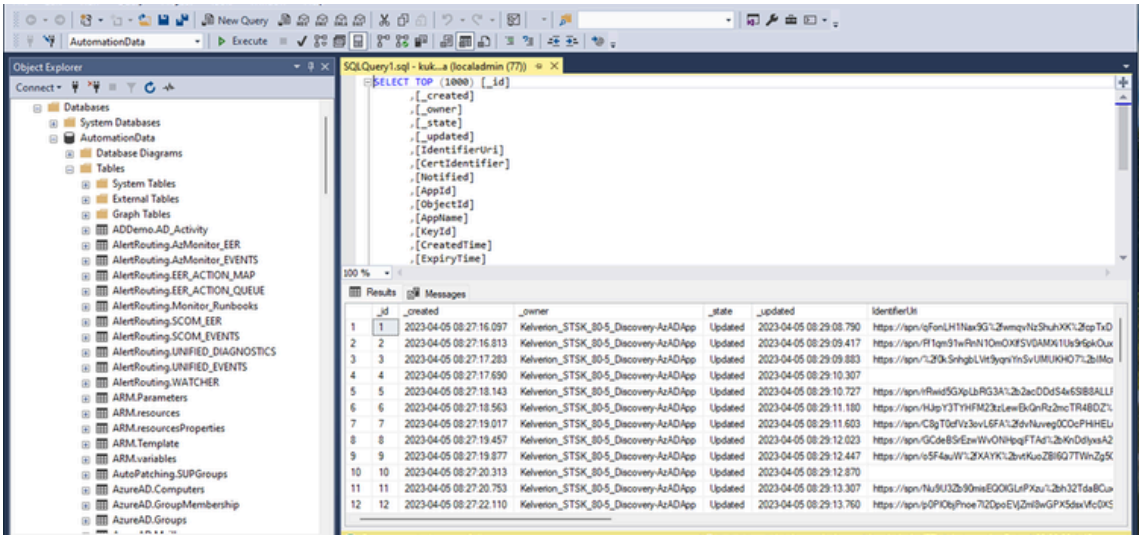
Persistence

Azure Automation runbooks will primarily use a PowerShell workflow. Each runbook is extremely powerful but runs as a standalone container, so Azure Automation does not know what was done in any other runbook, which means it cannot remember or keep track of what has been done before without having a central location to store the data that can be recalled later.

This means that if the runbook fails without completing, or a hybrid worker was to reboot or crash while a runbook was running, all data about executing runbooks would be lost.

So, a complicated multi-step workflow that is counting on internal runbook data for its current state would need to be restarted manually by an administrator. That administrator would potentially need to take significant time to understand what parts of the workflowhad run successfully and what parts were still needed. If a persistent data store is used following Kelverion best practices, the last successful step in the process would have been recorded in the database with all necessary information to continue the next step. This provides a basic audit trail of what has been done so the administrator can easily pick up where things left off after the issue has been addressed.

Fig 1: Kelverion's SCOM 2022 Connector PDS



Passing Data Efficiently Between Runbooks

Information gathered and generated by one runbook is not available to other runbooks in Azure Automation without implicitly passing that data to the child runbook. In that method, only defined runbook inputs can be passed to the child runbook. This means passing large amounts of data between runbooks is a time consuming and error-prone process. By writing the data to a SQL database at the end of each runbook, the data can be passed by simply supplying the child runbook the primary ID of that row in the database. This also allows runbooks to act on data collected from multiple other previous activities. In the case of a step that sends out a final status email, it would only take a pull from the database to retrieve the status of all the previous steps and include their data in the notification.

Fig 2: Passing Data between Runbooks Directly

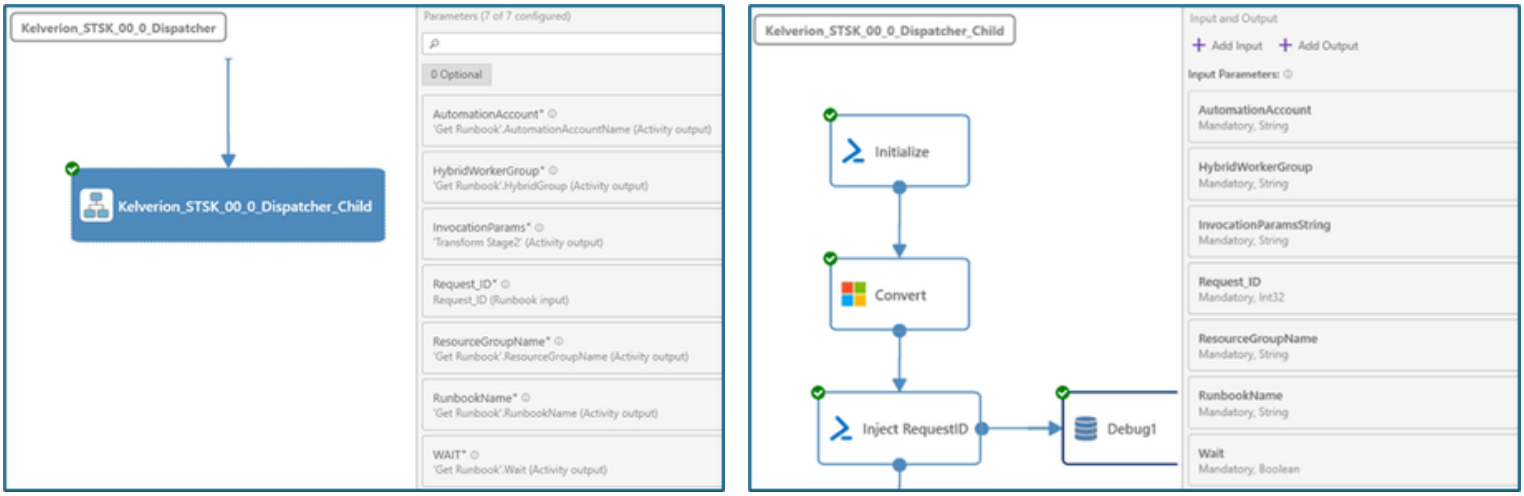
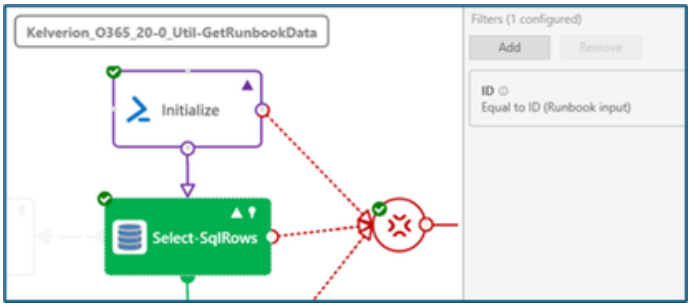
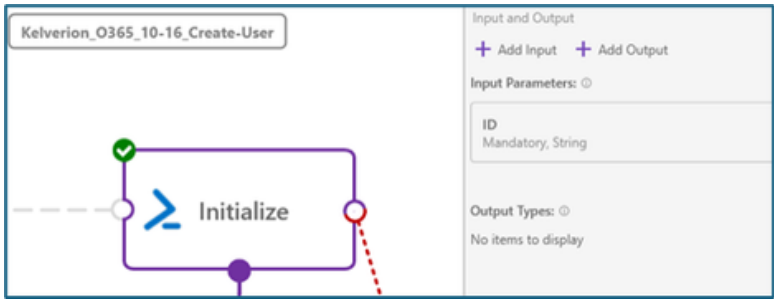


Fig 3: Passing Data Using a Persistent Database





By writing the data to an SQL database at the end of each runbook, the data can be passed by simply supplying a secondary runbook with the primary ID of that row in the database. This also allows runbooks to act on data collected from multiple other previous activities.

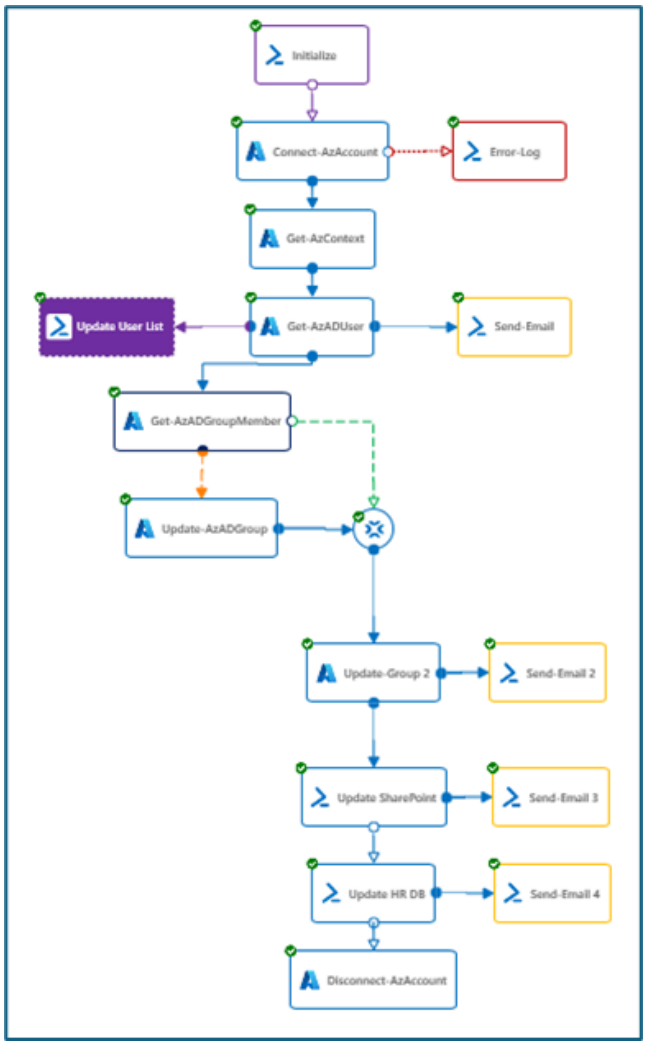
Senior Automation Consultant, Kelverion

Manageability

Because of some of the limitations inherent within Azure Automation, runbooks written without a database methodology tend to be large and harder to manage.

This is an example of such a runbook:

Fig 4: Example Runbook without a Persistent Data Store



By using a database in Azure, we can make smaller more manageable runbooks that work together in a modular fashion. These runbooks can handle smaller tasks and then pass the data reliably to the database, and then another concise runbook can pick it up from there. This makes it significantly easier to add, remove, or change an existing workflow.

Fig 5: Example Runbook using a Persistent Data Store

The positive experience that I had working with you both actually was mentioned in my statement of work presented to the management, as one of the deciding factors on why we should purchase products from Kelverion.

Senior Automation Analyst, Oil & Gas Company

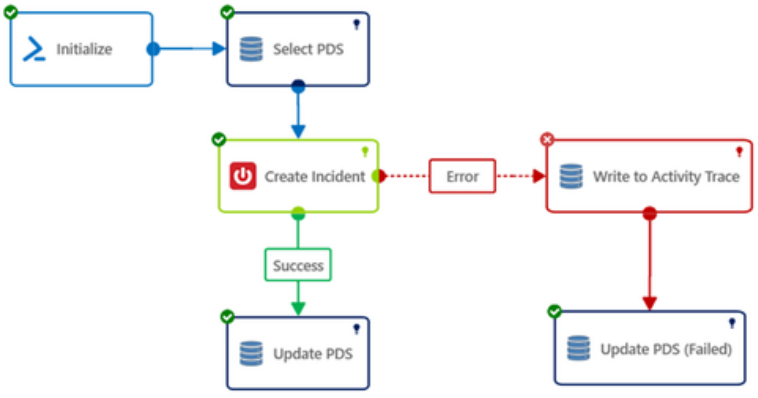
Rerunning and Troubleshooting Runbooks

The use of a persistent data store makes re-running a step in the process significantly easier.Because the data being passed is stored in the persistent data store, it is easy to access again without having to query the original source which means you save time and resources. It also means you can build more fault tolerance into your runbooks by showing which steps (or runbooks) have started and completed for that particular set of data. If we were to do this manually, we would not be able to track that level of detail and would most like have to start the process all over again if it failed somewhere in the middle. However, if we use a persistent data store where the data is stored in a database, all we would need to re-run the process is the “ID”.

Figure 6 below shows how Kelverion structures many of its runbooks when using our best practice by including a persistent database. The runbook input is receiving the ID of a table row from another runbook. The data from that table is then made available by the Select Rows activity.

In this example, the Select PDS activity from Kelverion’s Integration Module for SQL Server is used to get the data to create a new ticket. Kelverion’ s Update-SQL Row activity is used to record the success of the task. Additionally, any errors are recorded back to a different SQL table as well as updating the PDS record.

Fig 6: Example Best Practice Runbook

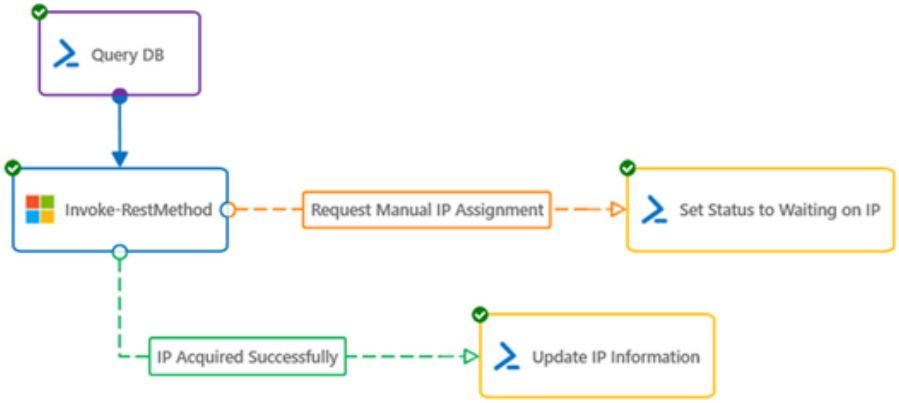


Waiting & Manual Intervention

Without a way of storing information outside of the runbook, the original data and any new data collected as the current runbook executes must be passed on to the next runbook immediately. Many processes gather data that needs to be used later, not just the next step. Additionally, some steps in the process need to wait for an approval or manual intervention. Because all the data needed to execute a runbook step exists in a persistent database; a step can be run reliably now or at any time in the future.

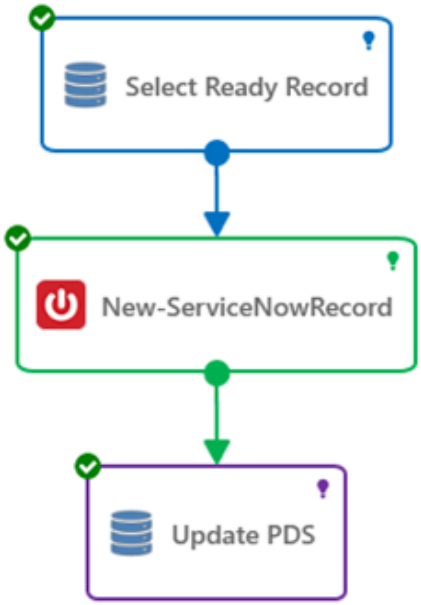
The runbook in figure 7 starts when the **Query DB** activity collects some data. Usually, this is looking for a status column in a table to be equal to a specific value. This runbook then uses a REST API call to request an available IP address to assign to a new server. If the IP address is provided, then it gets recorded back to the database along with a new status that will kick off the next step. If the IP address is not provided the process does not need to fail. The status of the process is set to a value that will in turn run a process that notifies the appropriate team that an IP address is needed and then once the IP address is provided the process can pick up where it left off.

Fig 7: Runbook Example with Manual Intervention Option



The manual intervention could be handled through the company's ticketing system. In Figure 8, the information from the PDS is used to create a ticket and request an IP Address from the correct team.

Fig 8: Create a Service Desk Ticket



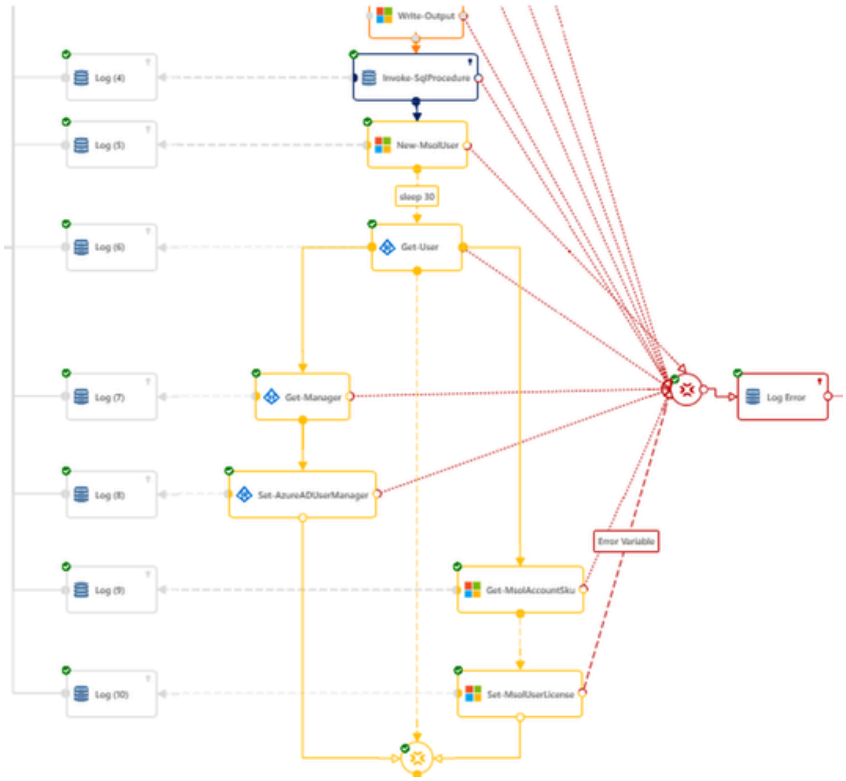
A runbook would then monitor that type of ticket for closure. The closed ticket should contain the new IP address. The address will then be written into the database with an updated status. That status would be the same as if the runbook in figure 7 had successfully found an IP. Whatever runbook would normally come next would now execute.

Fig 9: Monitor for Closed Ticket

Error Handling

Error handling is one of the most critical aspects of good runbook design. Trying to handle errors in each runbook would add a significant amount of work and potential rework. However, by creating an Alert Handling database table, error information can be written from any runbook to this table. Administrators can be notified using a single runbook that looks for new entries in the Alert Handling table. This table would have historical error information that can help identify error trends in the workflow. Figure 10 shows an example of this approach. You should also note, this approach can be used to account for multiple possible errors received and directed on to another process or runbook to resolve the error, then return to where it left off.

Fig 10: Error and Debug Logging



Speed of New Development

The process of developing new runbooks can be a time-consuming process. One of the primary metrics to evaluate when deciding whether a certain automation project should be done is the amount of time it will take to build and maintain compared to the amount of time saved. Kelverion has found that using this methodology involving a persistent database with a modular approach to runbook design in combination with our Integration Module for SQL Server reduces the overall time required to develop enterprise class workflows in Azure Automation by two thirds. If it takes substantially less time to create, maintain, and update workflows many more tasks can be automated. This results in an IT department that provides services more quickly, with less opportunity for human error thereby ensuring greater consistency, all combined for less cost and substantial savings.

"You guys are rock stars. I always got immediate and efficient support from both of you and really enjoyed working with you on automation.

Senior Automation Analyst, Oil & Gas Company

Kelverion Integration Module for SQL Server

The methodology described above is made most possible by the Kelverion Integration Module for SQL Server. The Kelverion Integration Module for SQL Server provides significantly improved capabilities and an enhanced user experience over writing native SQL integrations via PowerShell in Azure Automation Runbooks. Using Kelverion smart discovery the Integration Module interrogates the SQL instance and discovers the databases, their structure and properties. The Key capabilities added to your Runbook Studio solution include:

- Automatically builds and executes the necessary SQL commands without the user having to write or understand PowerShell
- Simplifies Runbook design by automatically mapping table columns to input properties, filters and published data items
- Prevents errors by controlling access to read-only fields, enforcing mandatory inputs and by providing value browsers for common data types.
- Smart Connections establish a reusable link between the Runbook Studio and a specific SQL Server database. You can create as many Smart Connections as you require, specifying links to multiple databases. You can also create multiple Smart Connections to the same database to allow for differences in security privileges for different user accounts.

The method of using an Azure database for runbookdesign can be partially accomplished with custom PowerShell scripts but with significantly less efficiency and almost none of the time savings. This would also defeat the benefits of low code/no code features available in our Integration Modules.

Kelverion's Integration Module for SQL provides you with the following activities:

The Kelverion Integration Module for SQL Server is included free with the Kelverion Runbook Studio. These are only a few of the advantages of Kelverion's Integration Module for SQL Server. Please find out more about the power of the IM for SQL Server and request an evaluation copy at www.kelverion.com.

"Kelverion has a lot of integrations that we use to make automation easier in our runbooks; we have not seen this amount of integrations covered by one company or done as well."

IT Administrator, Paint and Glass Manufacturer

About Kelverion

Experts in Cloud, On-Premise and Hybrid automation, Kelverion provide solutions and integrations that remove the manual process tying up IT staff; transforming the productivity, efficiency, and supportability of IT service automation. Our products utilize and enhance the power of Microsoft Azure and System Center Orchestrator.

Working closely alongside Microsoft, we have developed our integrations and automation solutions to help bridge the gap between Microsoft's automation platforms and third-party systems, in the process building key alliance partnerships with multiple vendors to ensure our products are fully certified.

Since 2010, Kelverion has expanded to become a global company, with offices now in the UK, Canada, and the US. Through this, we are able to offer and support products and professional services engagements to enterprise-level organizations no matter where they are.

There's a smarter approach to IT Service Management

Get in touch to find out more

www.kelverion.com

info@kelverion.com

US: +1 289 801 0559
UK: +44 203 875 8035

https://www.linkedin.com/company/kelverion-automation/