P R O C E S S M A N U A L

Document Code: PM-MSI

Microsoft Sentinel Integration

Send Auditlog data from your User Portal to your Microsoft Sentinel setup.

FastTrack Software

Admin By Request

Table of Contents

Introduction

Microsoft Sentinel offers various ways to consume data from different sources. As Admin By Request provides a public REST API for pulling Auditlog data (see the documentation here), it's an easy task to leverage the power of Azure Logic Apps to consume the Auditlog API and forward each new entry to an Azure Log Analytics Workspace for further Sentinel consumption.

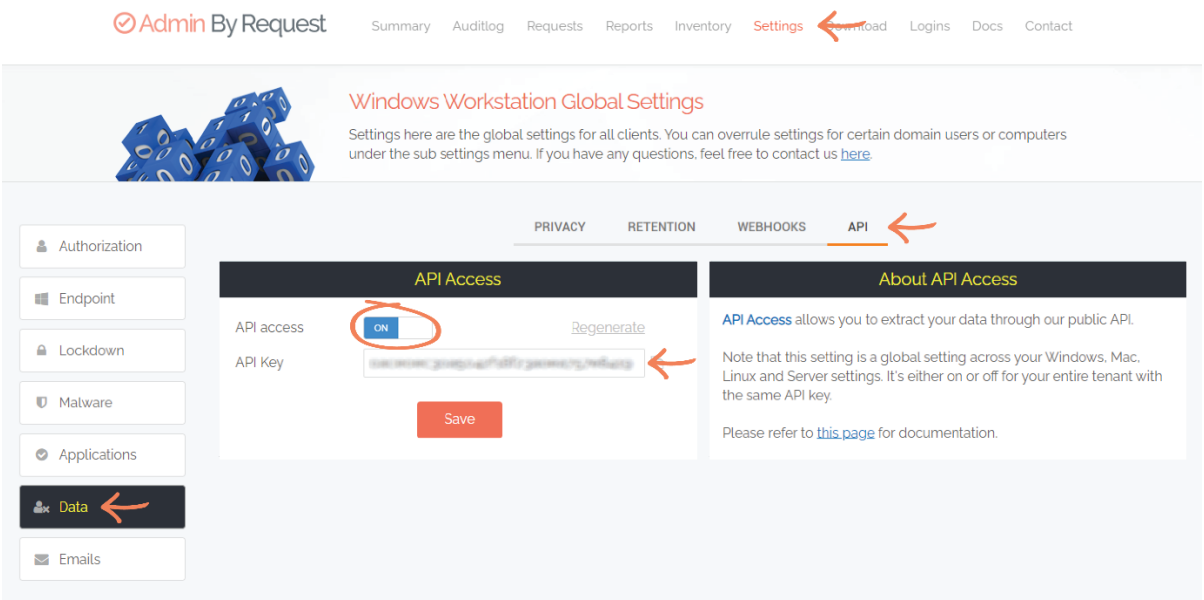
We've created an Azure Logic App that requires very few changes before having you up and running with Admin By Request Auditlog data in your Microsoft Sentinel setup. This manual provides a step-by-step guide on how to configure the integration.

Assumptions

The tasks described in this manual assume that the user has access to their Azure Portal, Admin By Request User Portal, and some familiarity with both environments.

Prerequisites

To enable this integration, you must first obtain your Admin By Request API Key. This key can be self-generated through your Admin By Request User Portal via **Settings > [OS] Settings > Data > API**:



IMPORTANT: Click the **Save** button after Regenerating an API Key, to ensure this is the key used to establish the connection to Azure. A green tick icon will appear next to the **Save** button when the action is complete:

NOTE: The API Key has been blurred out in the above example.

Breakdown of Tasks

Seven tasks are covered in this manual:

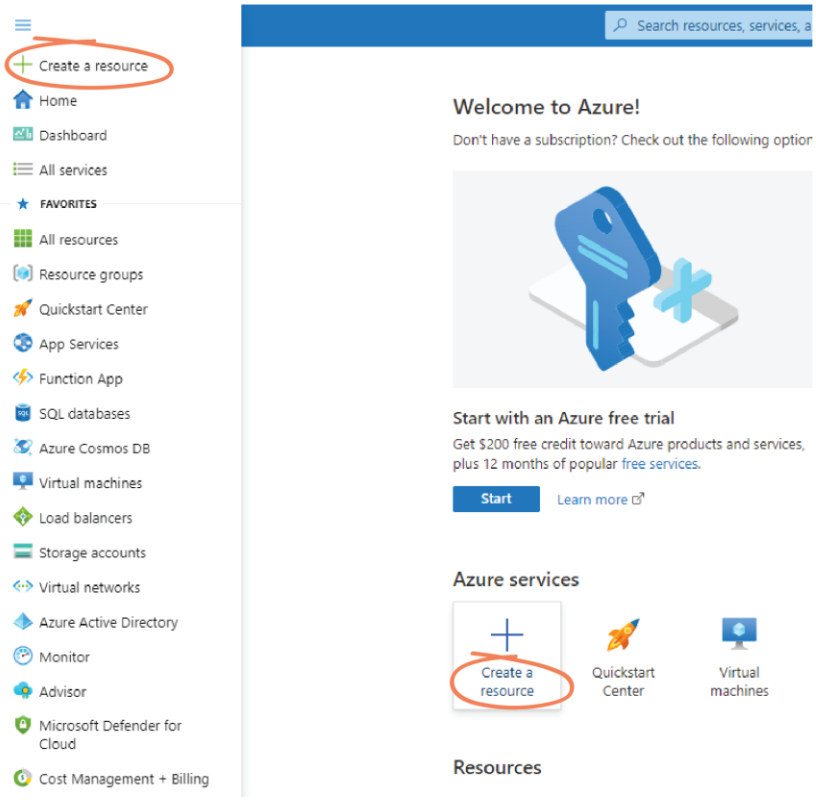
1. Task A: Set up Log Analytics Workspace
2. Task B: Create new Azure Logic App
3. Task C: Paste in JSON Code
4. Task D: Enter Parameters
5. Task E: Understand the App Flow
6. Task F: Configure Loop Entries
7. Task G: Test the Integration

Integration Tasks

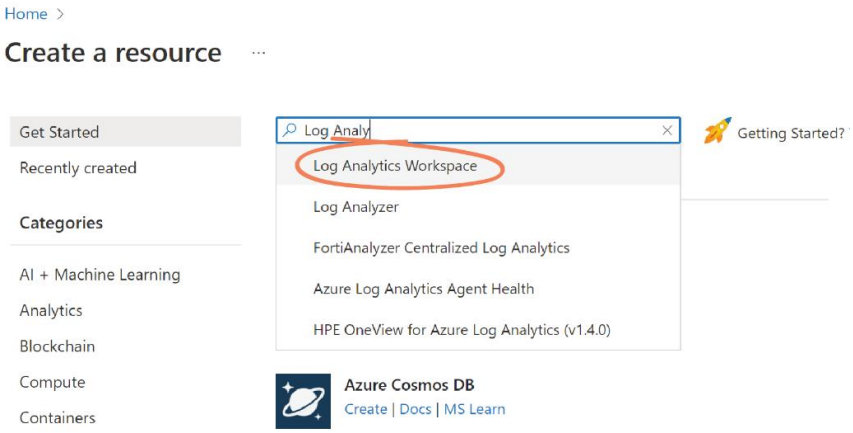
Task A: Set up Log Analytics Workspace

A Log Analytics Workspace is the management unit which allows you to store, query, and retain data pulled in from other tools – in this case, Auditlog data pulled from your Admin By Request User Portal. Task A involves setting up this storage unit for use in subsequent tasks.

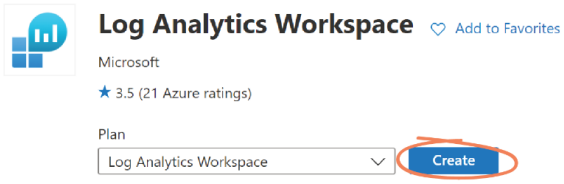
1. Log in to your Microsoft Azure Portal, and select **Create a resource** from the Home page or the side menu:



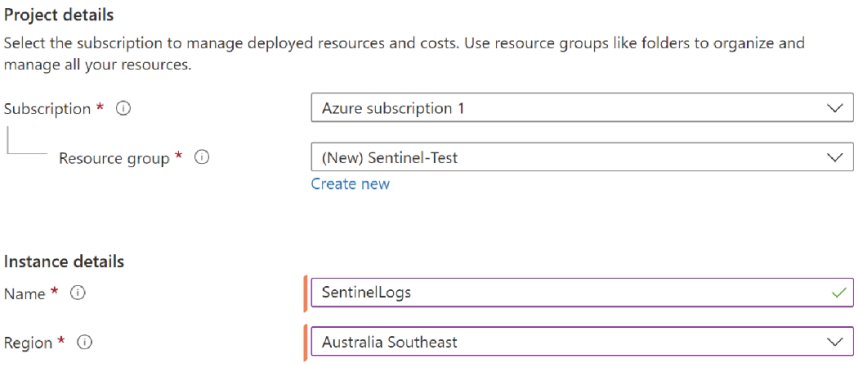
2. Use the search box to search for and select **Log Analytics Workspace** from the drop-down menu:



3. Select **Create**:



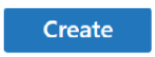
4. Fill out the Project details, and in the *Instance Details* section, give the Workspace a *Name* and select the appropriate *Region* from the drop-down menu:



NOTE: In the above screenshot, we have created a new Resource group called *Sentinel-Test* for the purpose of this demonstration.

5. Select the **Review + Create** button at the bottom of the page:

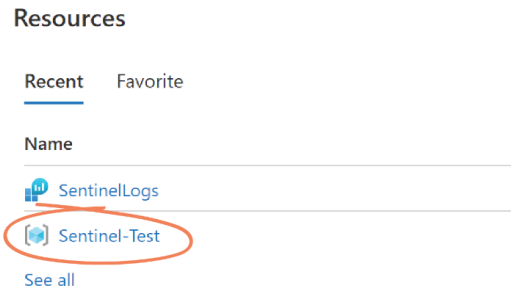
6. When validation has passed, select **Create**, and wait for deployment to complete:



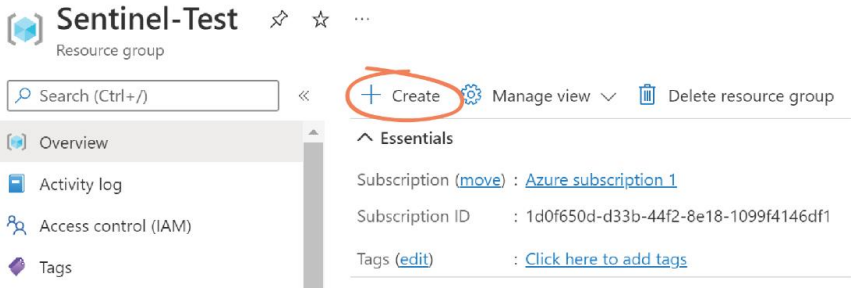
Task B: Create new Azure Logic App

An Azure Logic App is needed to consume the Admin By Request Auditlog API and forward each new entry to the Azure Log Analytics Workspace created in Task A.

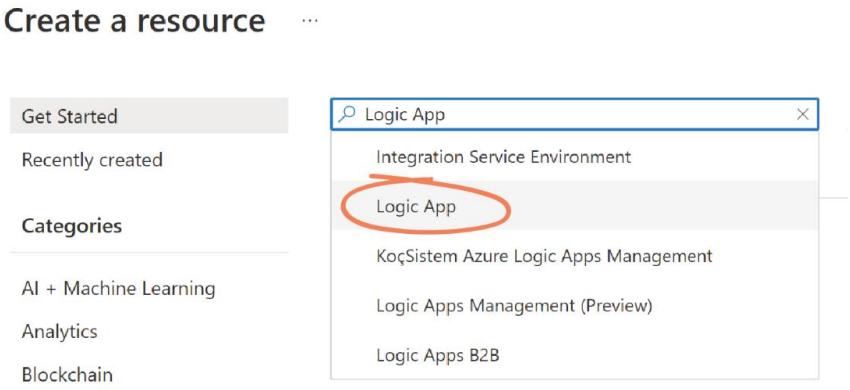
1. Navigate to *Resource groups* and select the Resource Group used in Task A from the *Recent* list under *Resources* – in this example, **Sentinel-Test**:



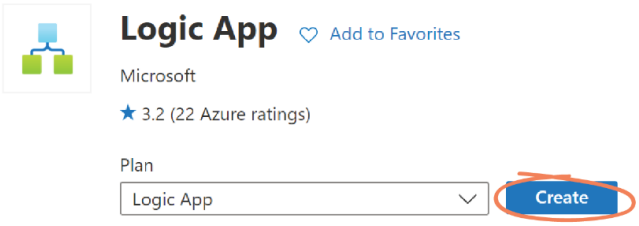
2. Once in Sentinel-Test, select the **Create** button:



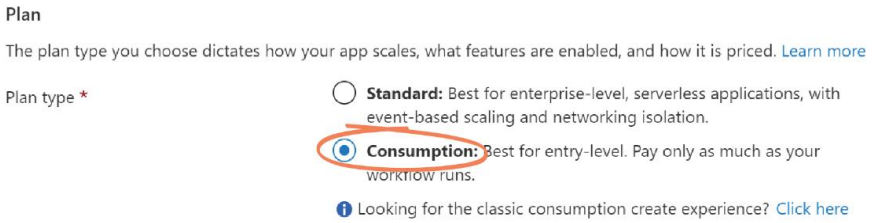
3. Use the Search bar to locate and select **Logic App** from the drop-down menu:



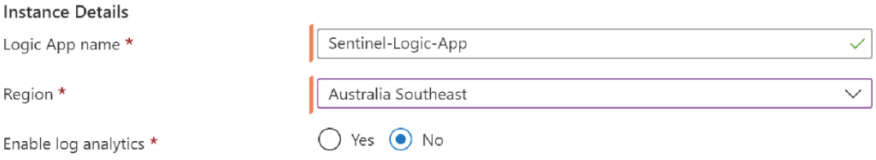
4. Click **Create**:



5. In the *Plan* section, select your *Plan type*. In this example, we use **Consumption**:

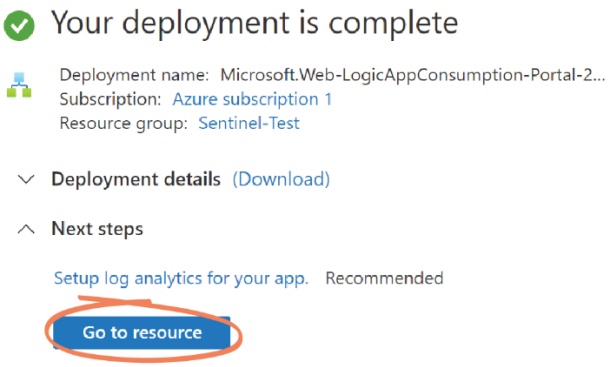


6. In the *Instance Details* section, enter a *Logic App name* (in this case, **Sentinel-Logic-App**) and select the appropriate *Region*:



7. Select the **Review + Create** button, followed by **Create**.

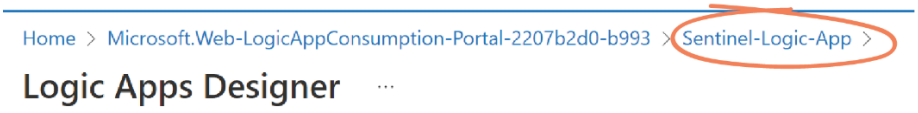
8. Once deployment is complete, click **Go to resource**:



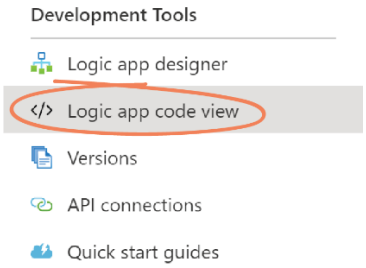
Task C: Paste in JSON Code

To get the app behaving how we need it to for this integration, you need to replace the default code in the Logic app code view with the JSON code we have written – download it here.

1. In the *Logic Apps Designer* page, select the app you created in Task B from the top menu (in this case, **Sentinel-Logic-App**):



2. From the left-hand menu, under *Development Tools*, select **Logic app code view**:



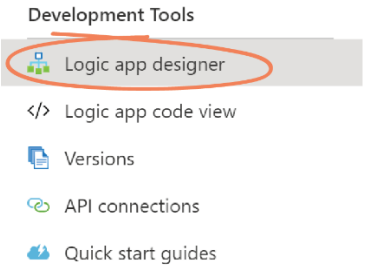
3. Open the Admin By Request JSON file (found here) and select and copy all code. Navigate back to the *Logic app code view* in Azure, and replace the existing code with the code copied from the JSON file:

4. Click **Save**:

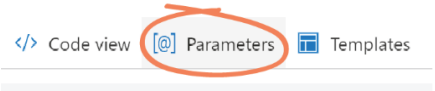
Task D: Enter Parameters

The Admin By Request API Key (found in the Prerequisite section of this document) is used to establish a connection between the Azure Logic App and your Admin By Request User Portal.

1. From the left-hand menu, under *Development Tools*, select **Logic app designer**:



2. From the top menu, select **Parameters**:



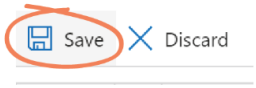
3. The two parameters required for the integration are:

   - *Apikey - String*: The API Key obtained from your Admin By Request User Portal (see Prerequisite).
   - *LogName - String*: The name you would like for the custom log in your Log Analytics Workspace.

   In the *Default Value* field for each of these parameters, replace the placeholder text with the appropriate / desired value:

NOTE: In the above screenshot, the API Key is blurred out, and we have used *AdminByRequestLogs* as the *LogName*.

4. Select the **Save** button in the Logic app designer:

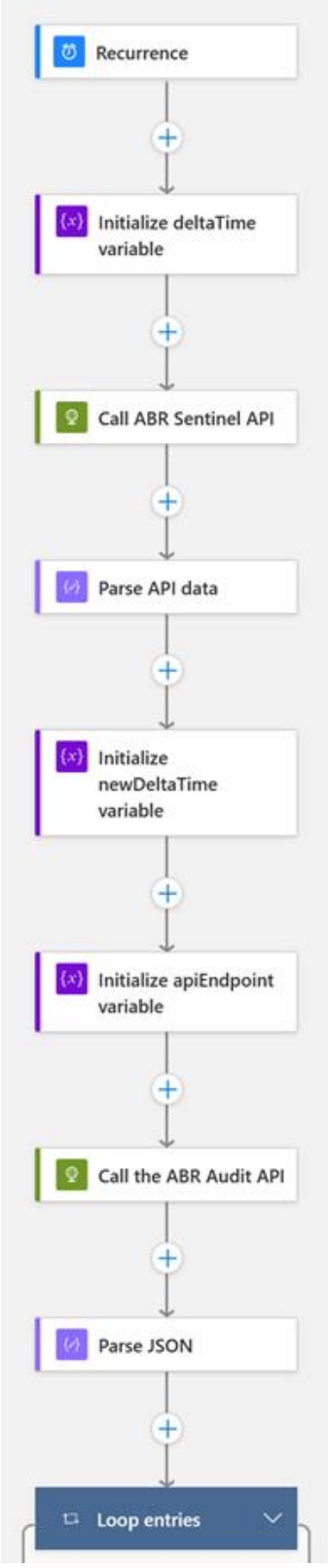


5. Close the Parameter window by selecting the **X** in the top right-hand corner.

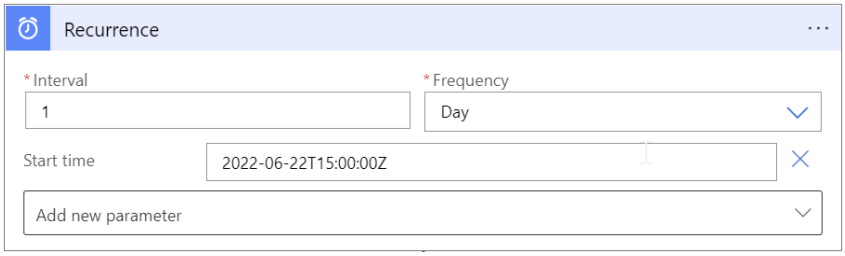
Task E: Understand the App Flow

In this Task, we take a look at what's going on 'behind the scenes' – at API calls, variables, and loops involved in the preconfigured app flow.

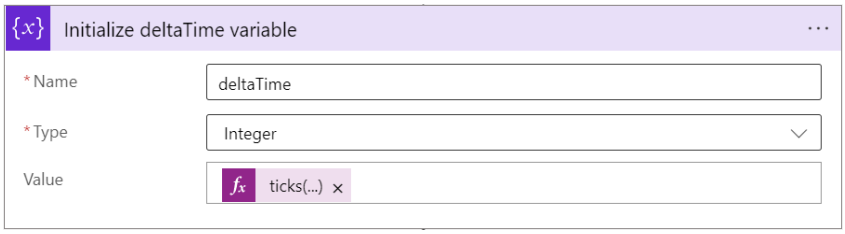
The app flow has nine segments arranged as follows:



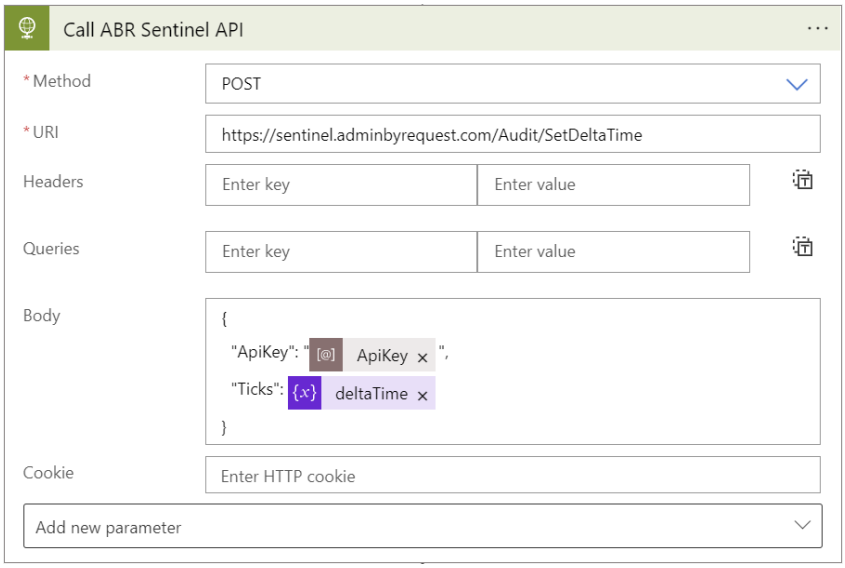
- **Recurrence** – This tells the app when it should run. In our example we've set up a recurring trigger that runs once every day. You can replace this trigger with whatever works best for your setup.



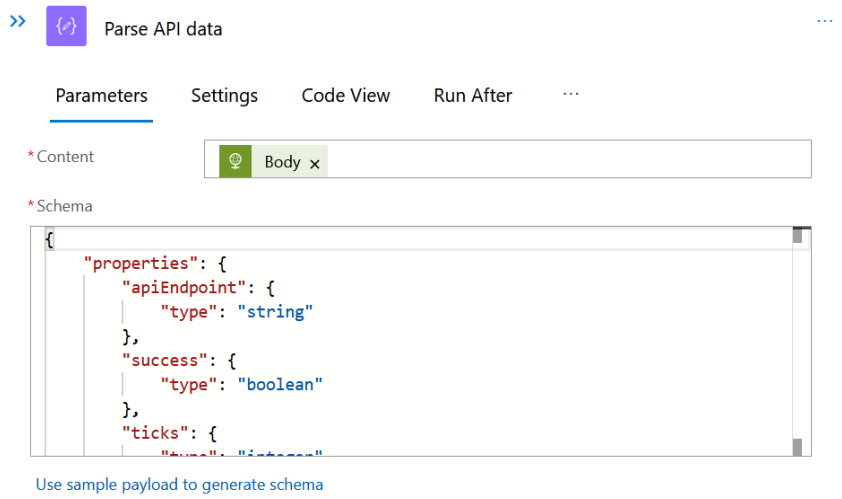
- **Initialize deltaTime variable** – In order to call the Admin By Request Audit API, we need a variable containing the 'from' ticks. Basically, telling the Audit API to 'give me all audit logs since this time'. This is defaulted to the number of ticks representing DateTime.Now.



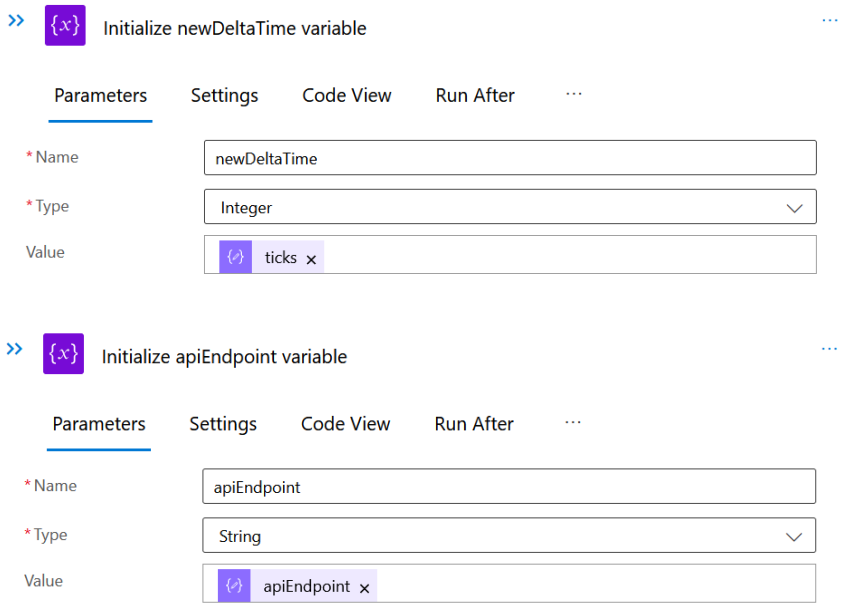
- **Call ABR Sentinel API** – Since Logic Apps don't hold any state, we need some way of storing the last time the Audit API was called for a given API-key. We've created an API endpoint that allows you to do just this. We simply call the SetDeltaTime endpoint with your API Key and the deltaTime variable, and the API returns that value for when the Audit endpoint was last called – and it stores the new value, so that the next time the Logic App runs, it has the correct tick-values to ensure that you don't get any duplicate entries.



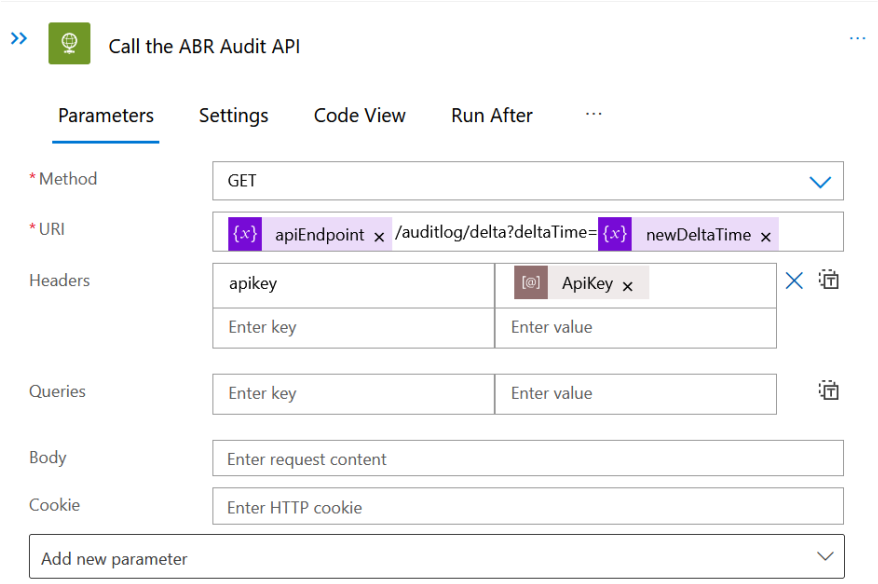
- **Parse API Data** – The result from the API needs to be parsed in order to use the resulting variables.



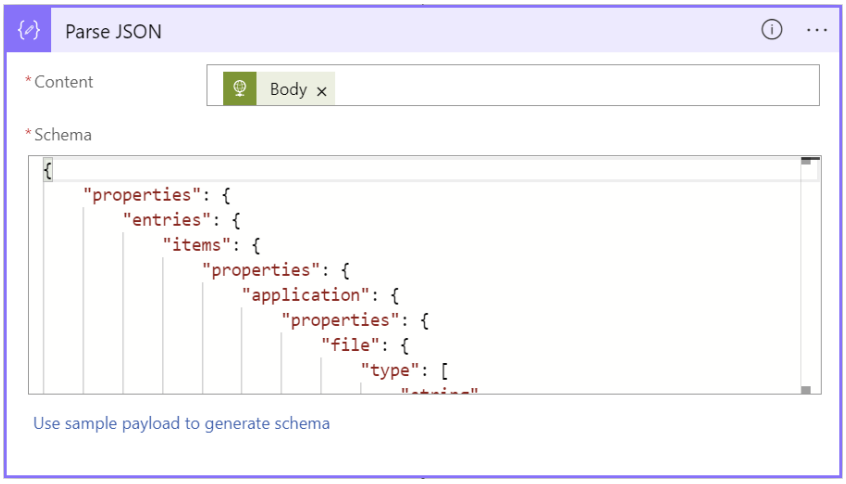
- **Initialize newDeltaTime and apiEndpoint variable** – With the values from the SetDeltaTime endpoint, we need to store two variables: newDeltaTime and apiEndpoint. These variables hold the tick-value for when the Audit API was last called, as well as the Admin By Request endpoint to call for Audit logs.



- **Call the ABR Audit API** – Now it's a matter of calling the Admin By Request Audit endpoint with your API Key, as well as the newDeltaTime variable.



- **Parse JSON** – The next step parses the response from the Audit API as JSON using a schema based on the response type from the Audit API (view the Auditlog API documentation for more information on this).

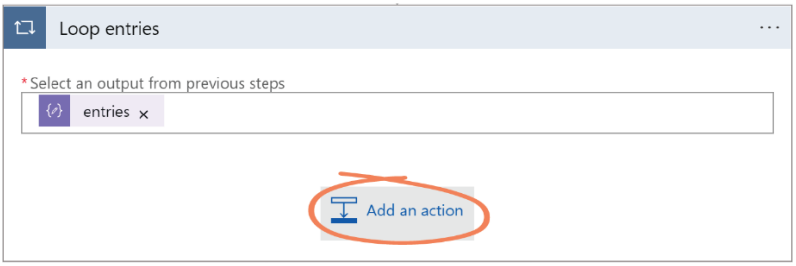


- **Loop entries** – The final step in the app flow simply loops through every entry from the Audit call. Here you decide what to do with the data. (See Task F, below.)

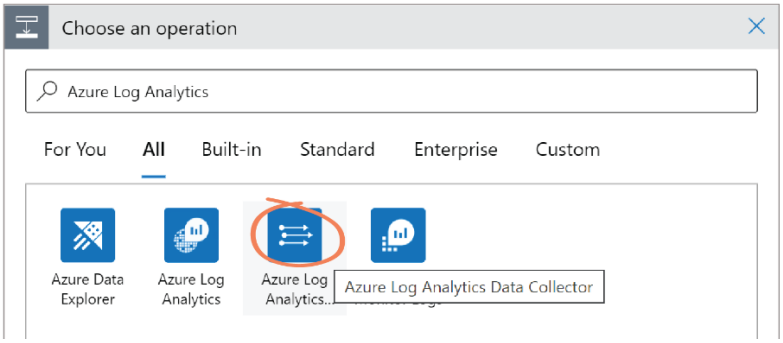
Task F: Configure Loop Entries

In order to send data to your Azure Log Analytics Workspace, you must add an action for each entry in the dataset.

1. Select the *Loop entries* segment of the app flow and click the **Add an Action** button:

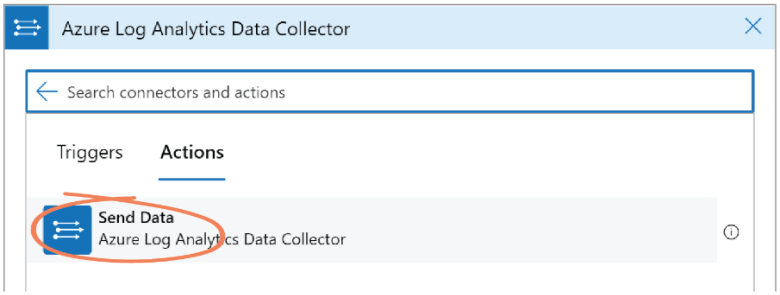


2. In the *Choose an operation* section, use the search box to locate and select **Azure Log Analytics Data Collector**:



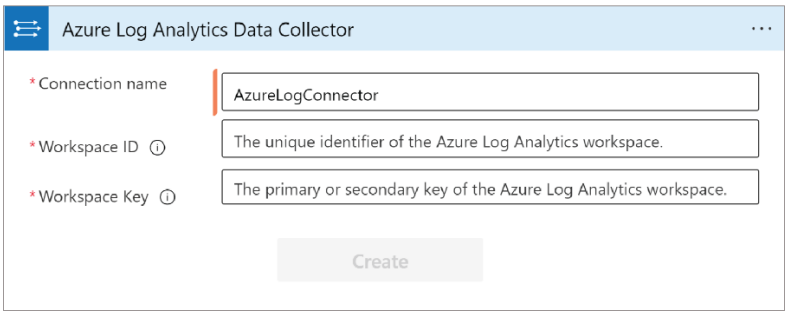
IMPORTANT: Be sure to select **Azure Log Analytics <u>Data Collector</u>**, rather than *Azure Log Analytics* (to the left in the screenshot above).

3. Select **Send Data** under the *Actions* tab:

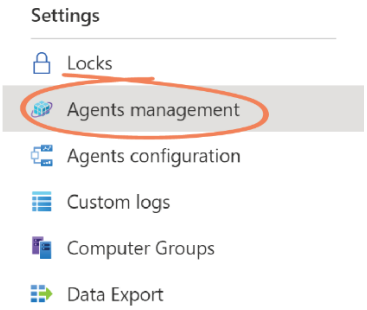


IMPORTANT: After clicking Send Data, you may be prompted to create a connection. If so, follow the steps below (steps 4-7). Otherwise, skip to step 8.

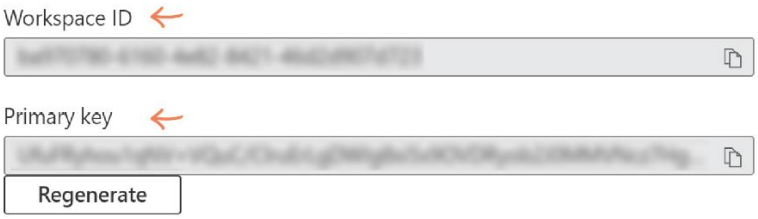
4. If prompted to create a connection, in the *Connection name* field, choose your desired name – in this case, we've used *AzureLogConnector*:



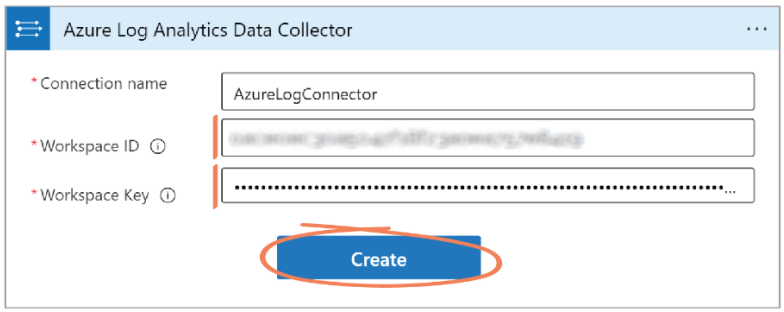
5. To locate the *Workspace ID* and *Workspace Key*, open your Log Analytics Workspace (i.e., *SentinelLog*) in a new tab and select **Agents Management** under *Settings* from the left-hand menu:



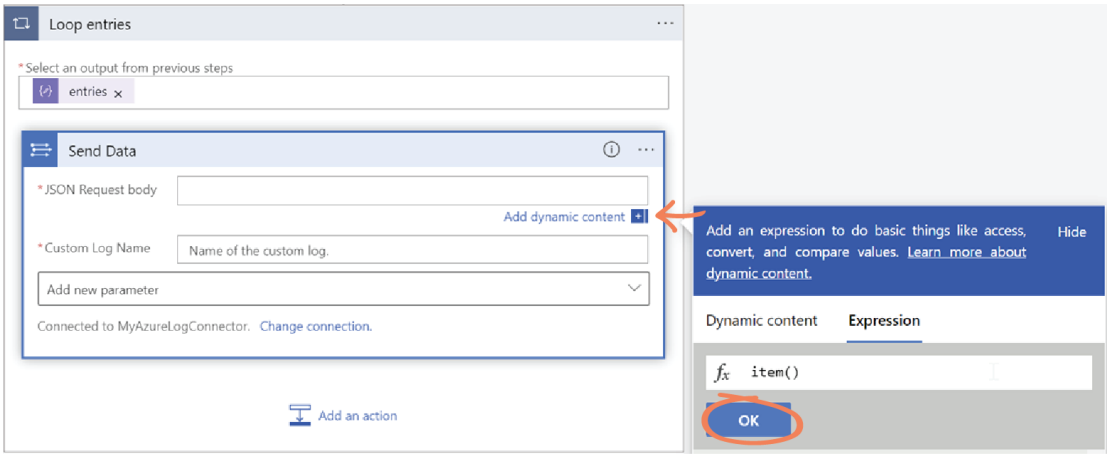
6. Copy the *Workspace ID* and *Primary key* values from this page:



7. Navigate back to the Logic App, paste the keys into their corresponding fields in the *Azure Log Analytics Data Collector* window, and select **Create**:

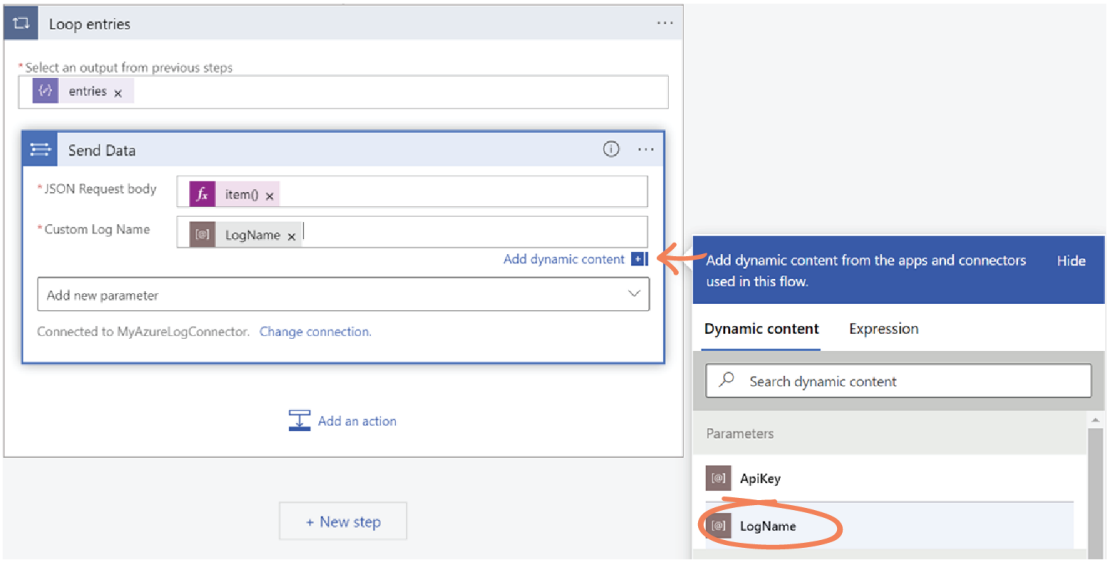


8. In the *JSON Request body* field, select **Add dynamic content**, and in the *Expression* tab, enter *item()* and click **OK**:



> **NOTE:** This selects the current item in the JSON loop and adds it as the request body.

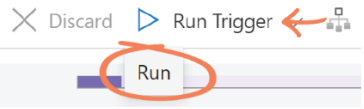
9. In the *Custom Log Name* field, select **Add dynamic content**, and in the *Dynamic content* tab, locate and select the **LogName** parameter:



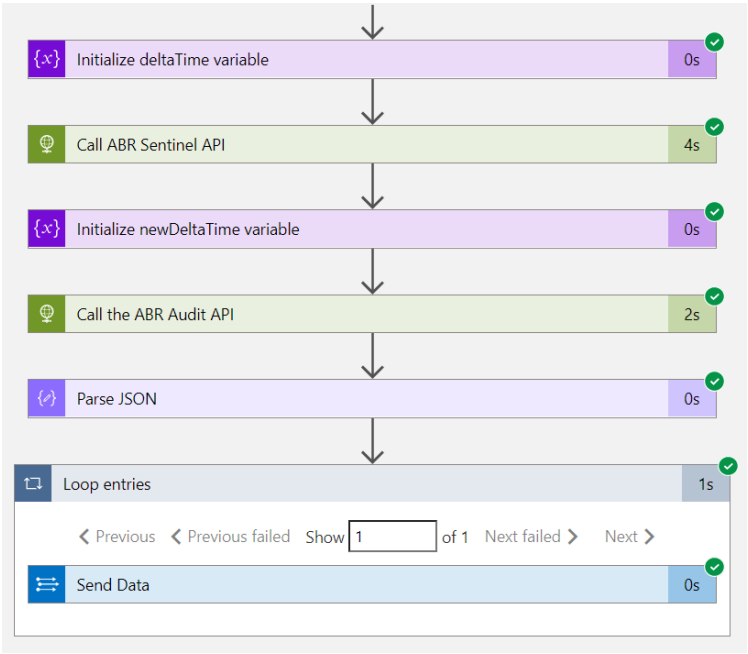
10. Select the **Save** button to save your app.

Task G: Test the Integration

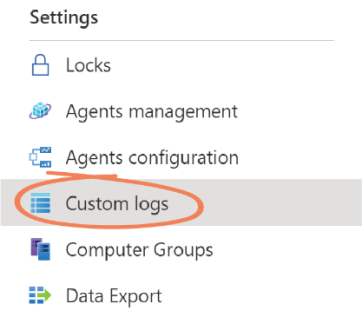
1. Select **Run Trigger > Run** from the top menu:



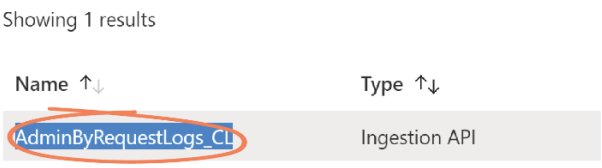
NOTE: You may need to wait a few minutes for the flow to complete. When successful, it should look something like the following:



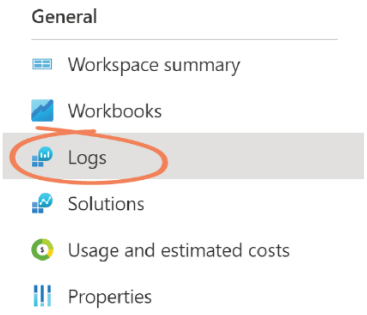
2. Navigate to your Log Analytics Workspace (i.e., *SentinelLog*), and select **Custom logs** under *Settings* from the right-hand menu:



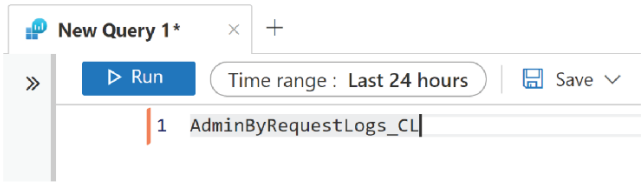
3. Highlight and copy the *Name* of the Custom log listed – in this case, *AdminByRequestLogs_CL*:



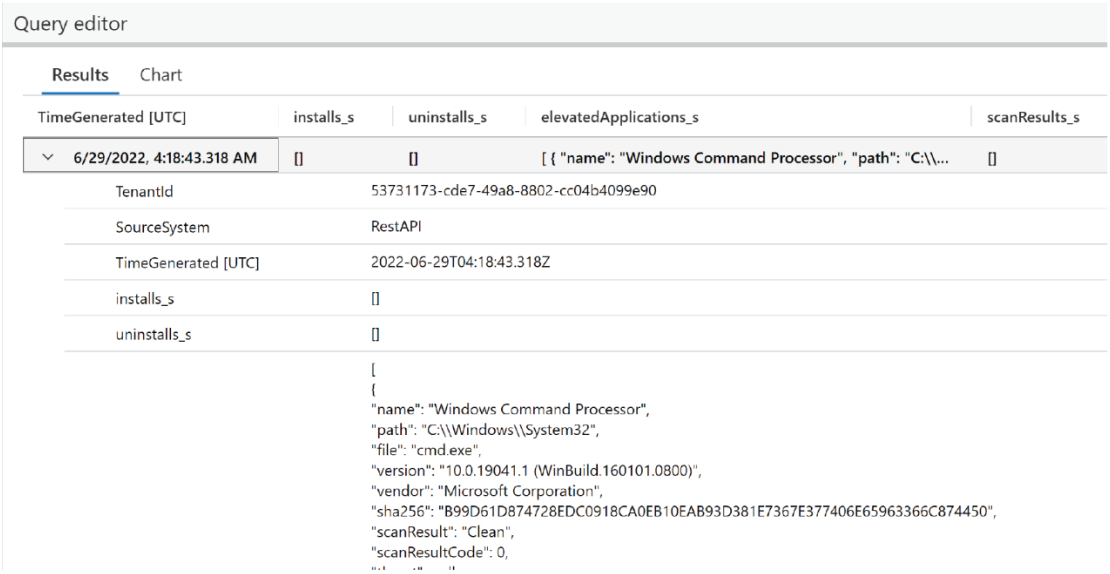
4. Select **Logs** under *General* from the left-hand menu:



5. Close the *Queries* window that pops up. Paste the copied Custom log in the *New Query* field and select **Run**:



6. New entries will begin to display in your Log Analytics Workspace as they are pushed through. Click the drop-down arrow to display details for each entry:



NOTE: It may take several minutes for log entries to show up in the Log Analytics Workflow.

7. Click **Save** to save the query for later use.

8. With the Azure Log Analytics Workspace set up, you can now point your Sentinel setup to use this workspace as a data source.