# Power Platform + AI: Get Future-Ready & Accelerate Innovation

##### **Lab 4: Implementing AI Search for Efficient Information Retrieval for IT staff**

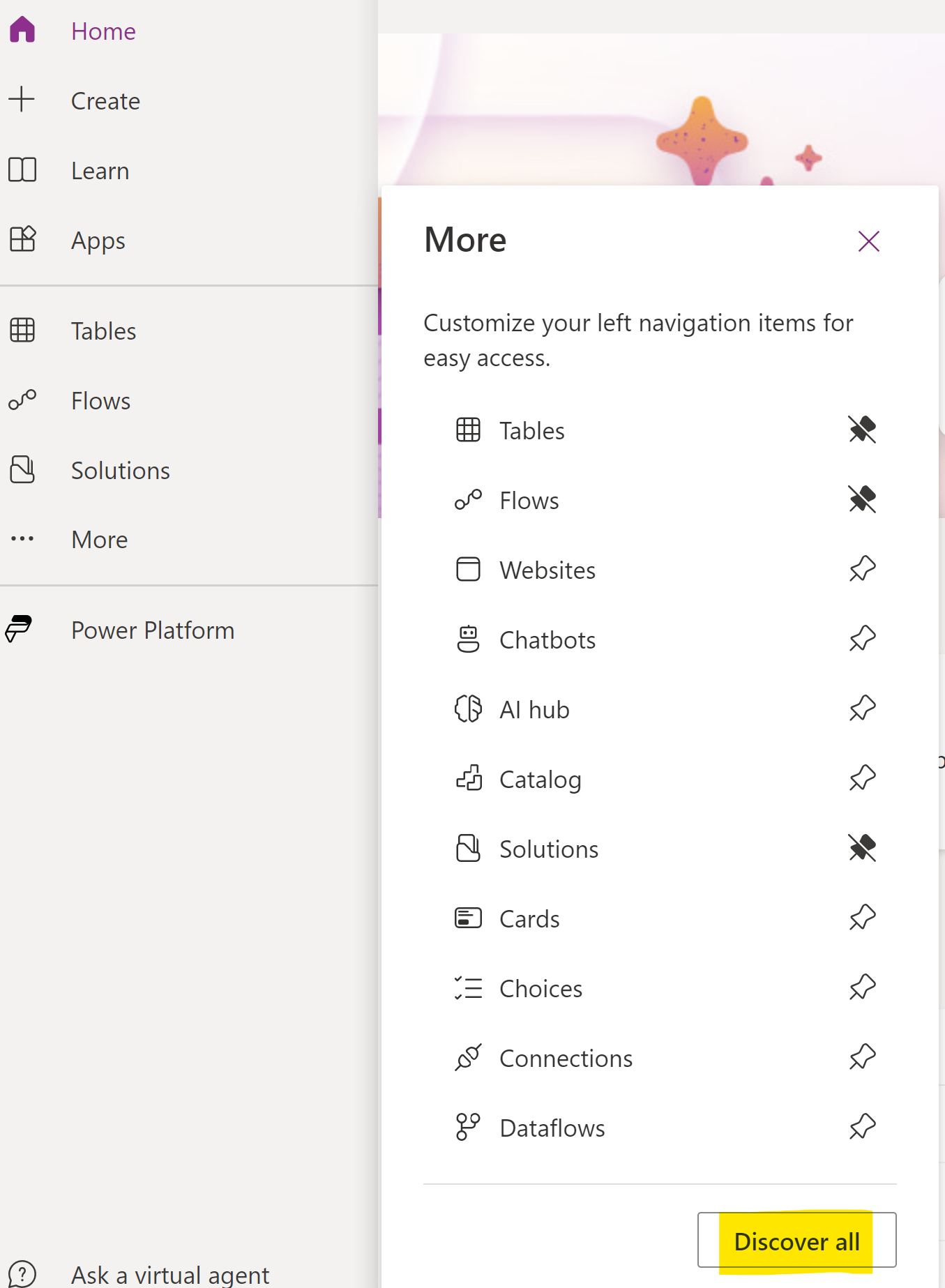
* **Objective:** Enhance the Help Desk app with AI Search to allow IT staff to quickly retrieve relevant information from knowledge base to reduce response time.
* **Tasks:**
  + Integrate AI Search to allow IT staff to quickly find similar past tickets, relevant knowledge base articles, and documentation.
  + Leverage an existing AI Search index (knowledge base in Azure SQL) that has been prepped for this lab and create a new custom connector to be used in Power Apps.
  + Add a Knowledge Base query screen in Power App to leverage the custom connector to pull relevant solutions via AI search and display in a gallery.
* **Outcome:** The Help Desk app will now include AI-powered search capabilities, enabling IT staff to quickly find relevant information and resolve issues more efficiently.

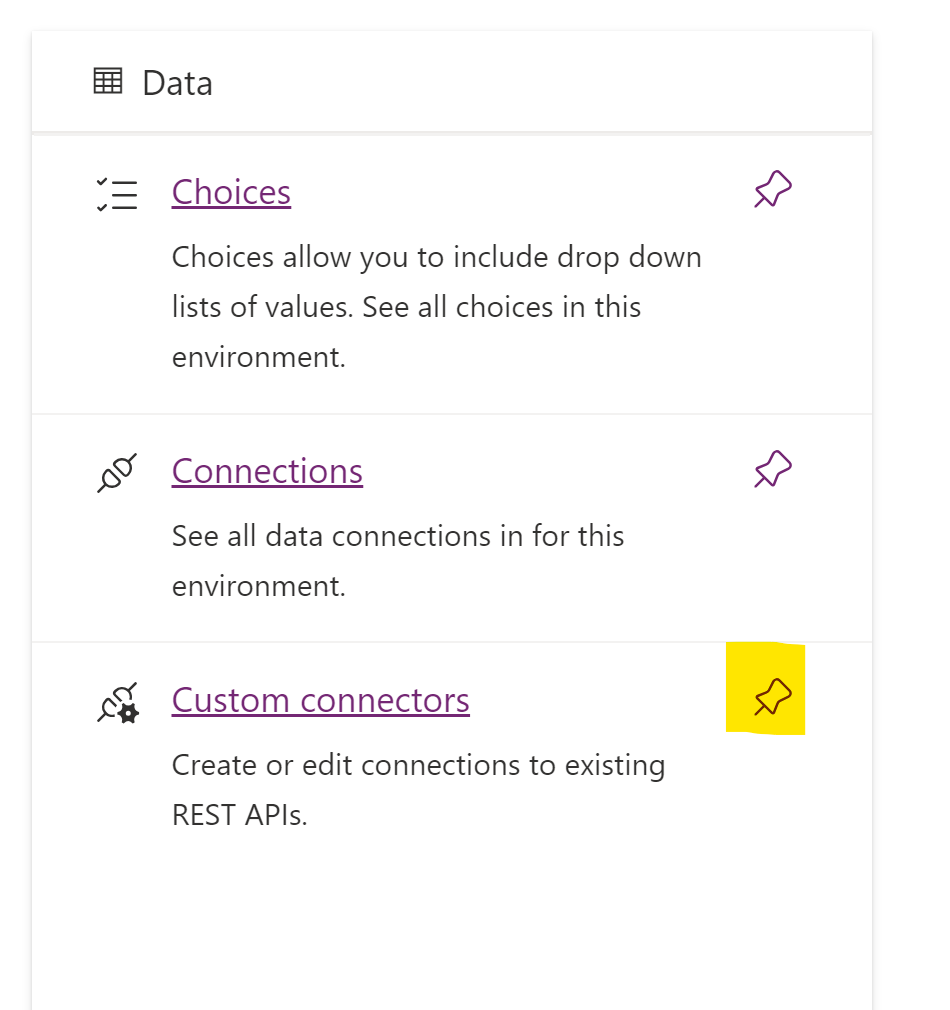
### **Create a custom connector**

Azure AI Search for Azure SQL Database is an intelligent search solution that combines the power of Azure Cognitive Search with Azure SQL Database. It allows users to perform advanced searches across structured and unstructured data stored in Azure SQL databases, using AI-powered capabilities such as natural language processing, semantic search, and text analytics. We have an Azure SQL DB set up with sample knowledge base of 10 historical ticket resolutions and a SQL search index created on an Azure subscription that you can leverage as part of this lab module.

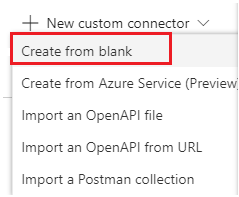
We will create a custom connector to query AI Search for relevant information and use the connector in the Power App so IT staff can quickly resolve issues. Knowledge Base data can be found in an excel sheet in the ‘Knowledge Base’ folder provided with the workshop materials.

1. Navigate to <https://make.powerapps.com/> > Select More on the left menu > Discover All > Scroll to the bottom of the page and click on the Pin next to Custom Connectors to pin it to the left menu.





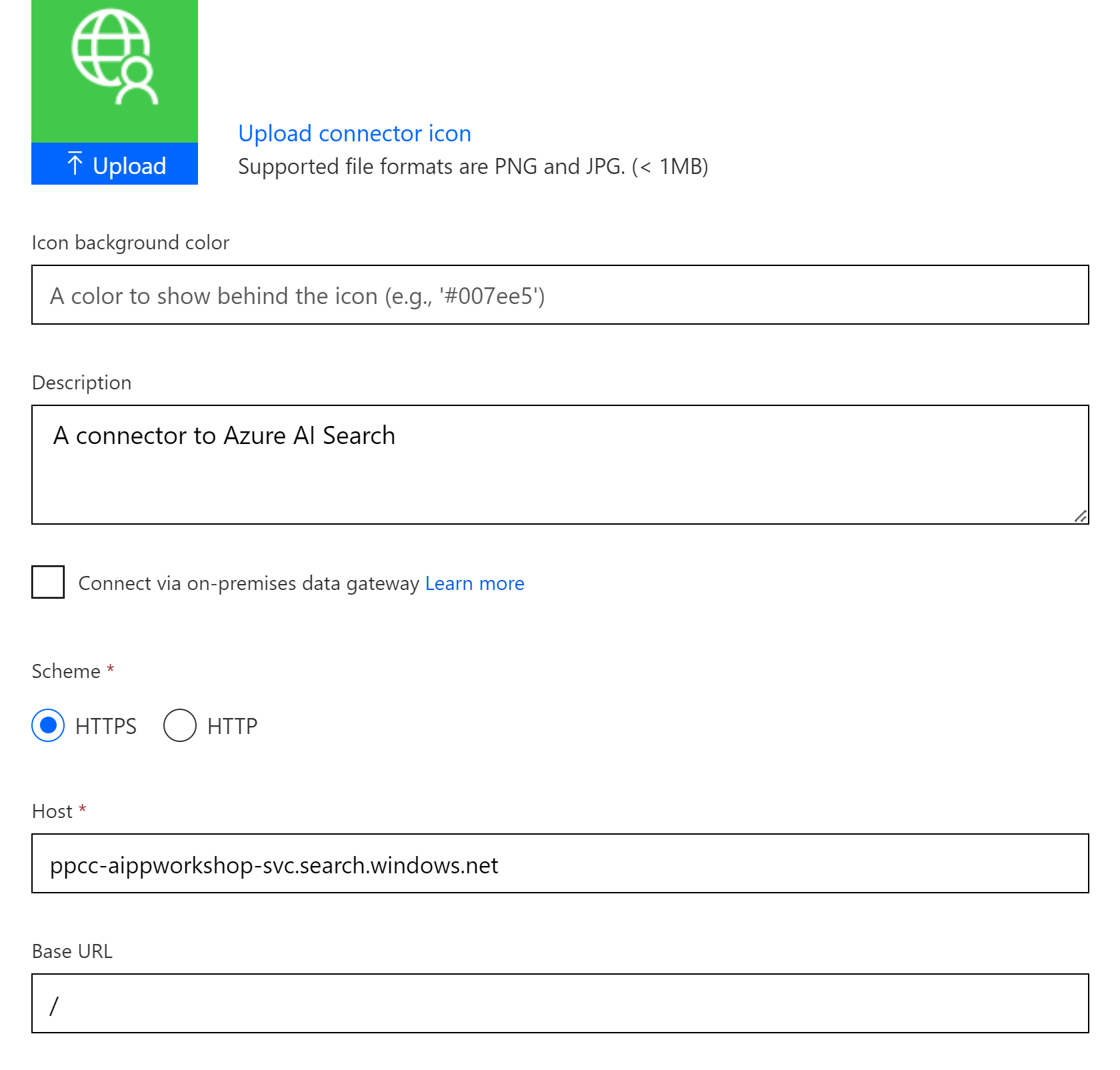
1. Select **Custom Connectors**. Select **+ New custom connector**, and then select **Create from blank**.



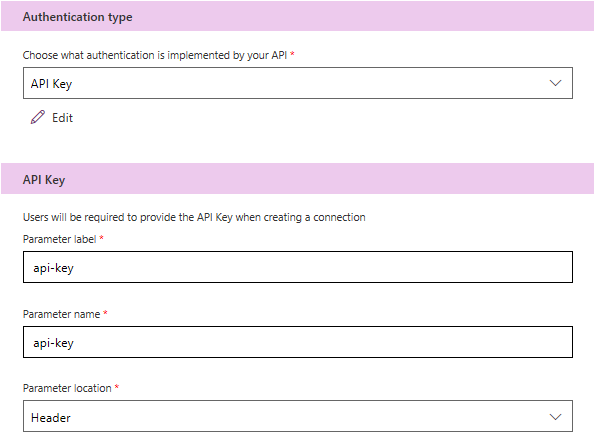
1. Give your custom connector a name (for example, *KBSearchQuery*), and then select **Continue**. Enter information in the General Page:
   1. Description (for instance, "A connector to Azure AI Search")
   2. In the Host, enter AI search service URL

*ppcc-aiworkshop-svc.search.windows.net*

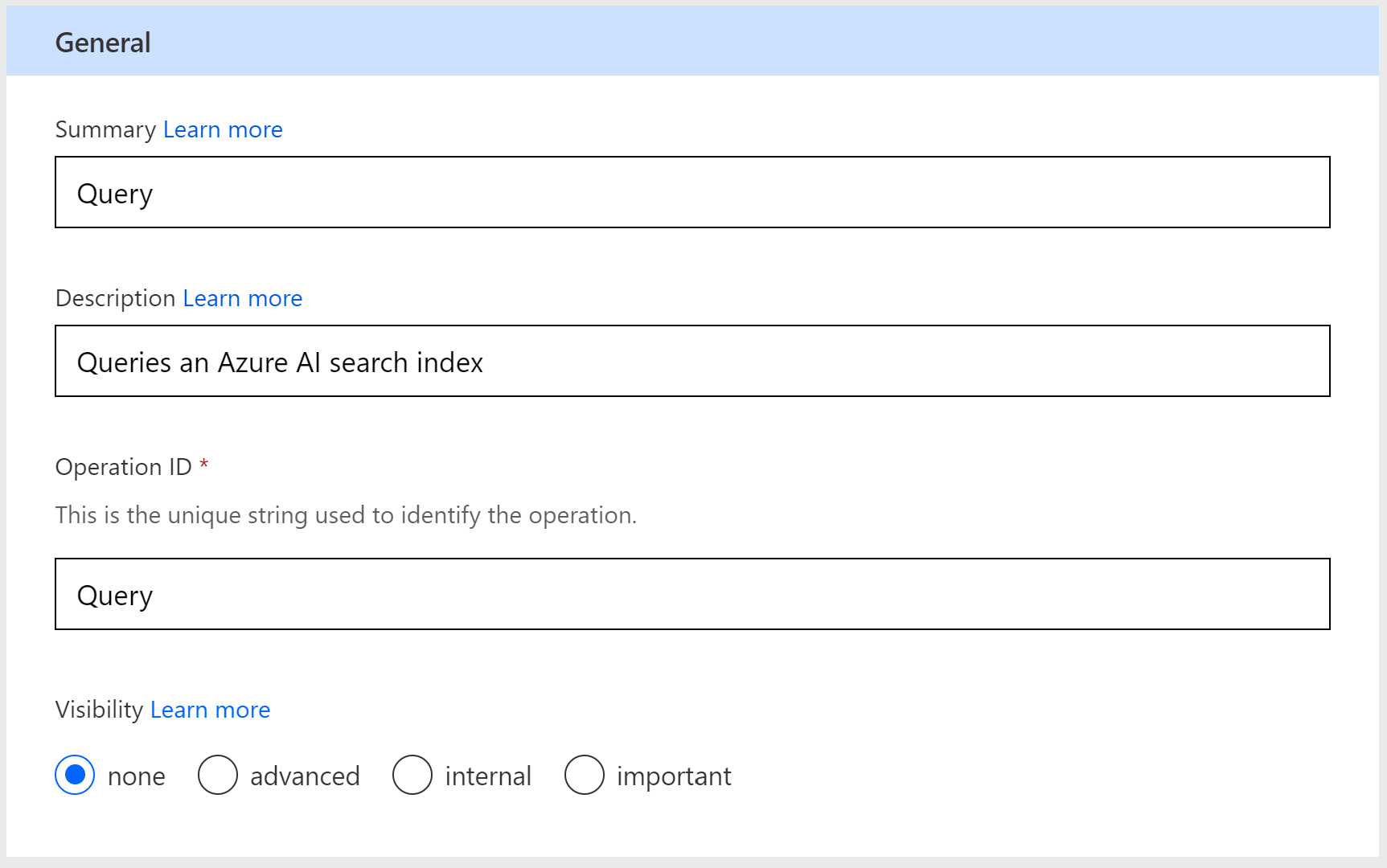
* 1. For Base URL, enter "/"



1. Click **Security ->** to viewthe Security Page, set *API Key* as the **Authentication Type**, set both the parameter label and parameter name to *api-key*. For **Parameter location**, select *Header* as shown in the following screenshot.



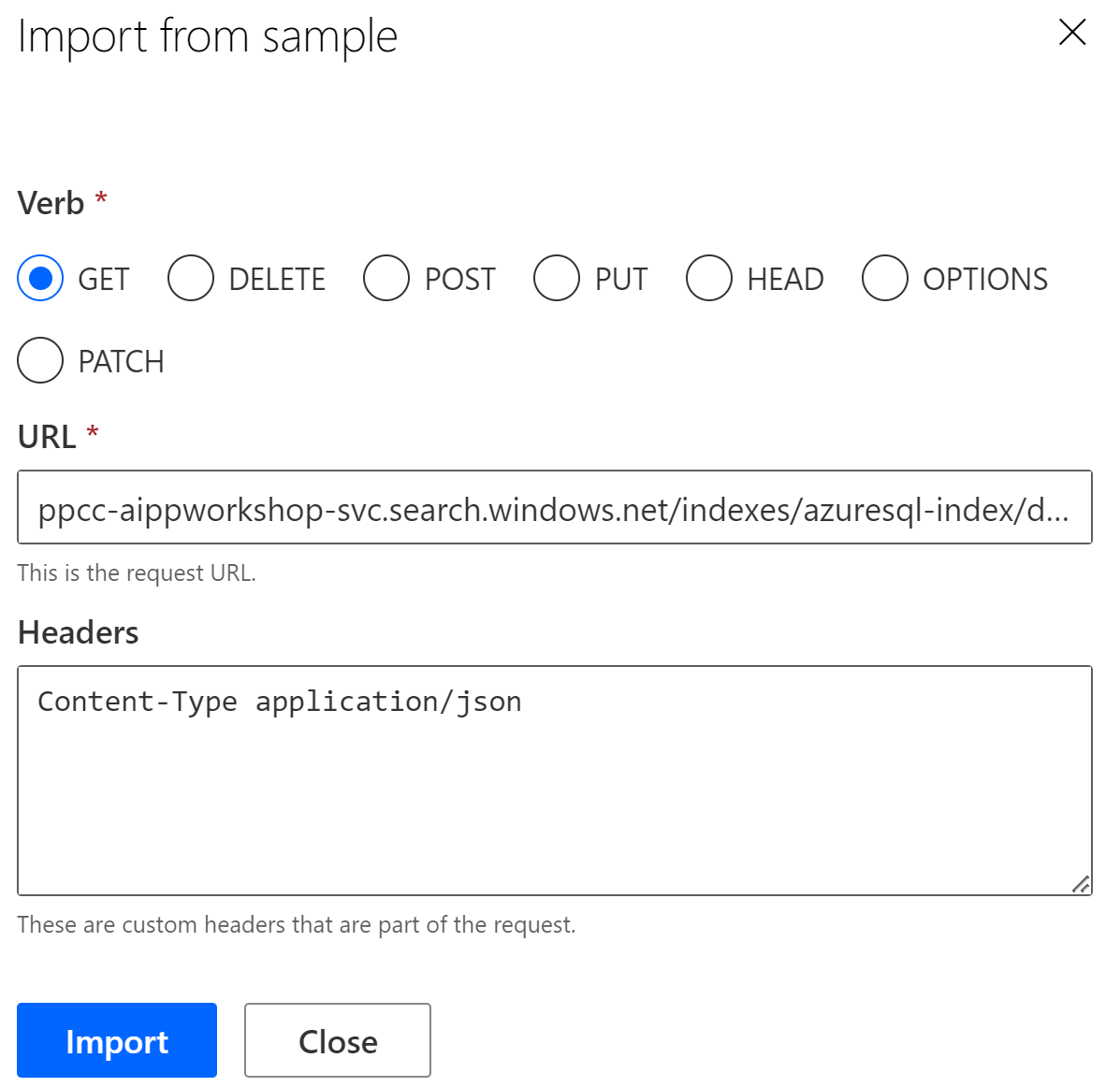
1. Select Next. In the Definitions Page, select **+ New Action** to create an action that queries the index. Enter the value "Query" for the summary and the name of the operation ID. Enter a description like *"Queries* *an Azure AI search index"*.



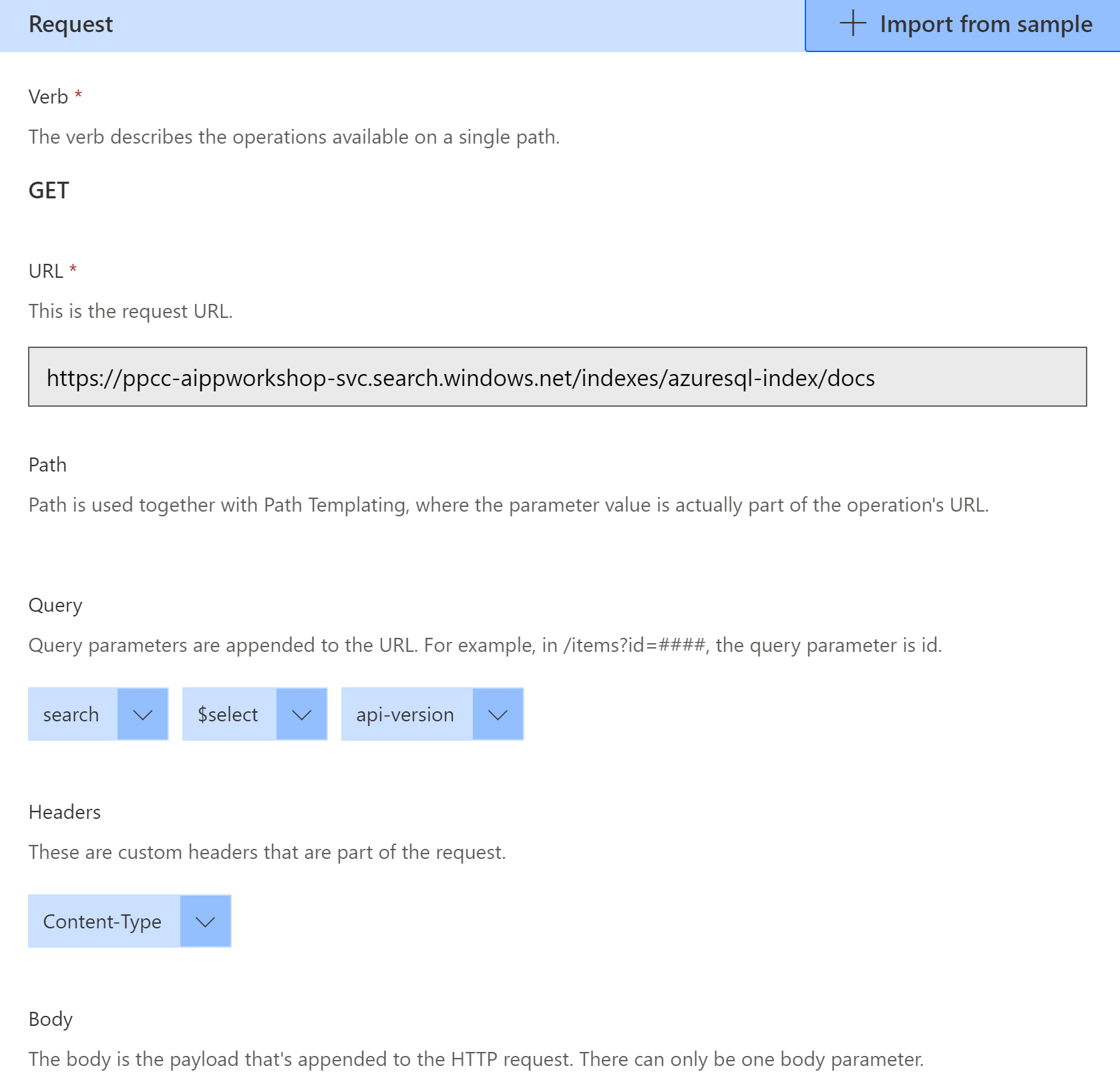
1. Scroll down. In Requests, select **+ Import from sample** button to configure a query request to your search service:
   1. Select the verb GET
   2. For the URL, enter a sample query for your search index (search=\* returns all documents, $select= lets you choose fields). The API version is required. Use URL below (Omit the https:// prefix.):

indexes/azuresql-index/docs?search=\*&$select=Issue,IssueDescription,ResolutionSteps,AssignedTo&api-version=2024-07-01

* 1. For Headers, type Content-Type application/json

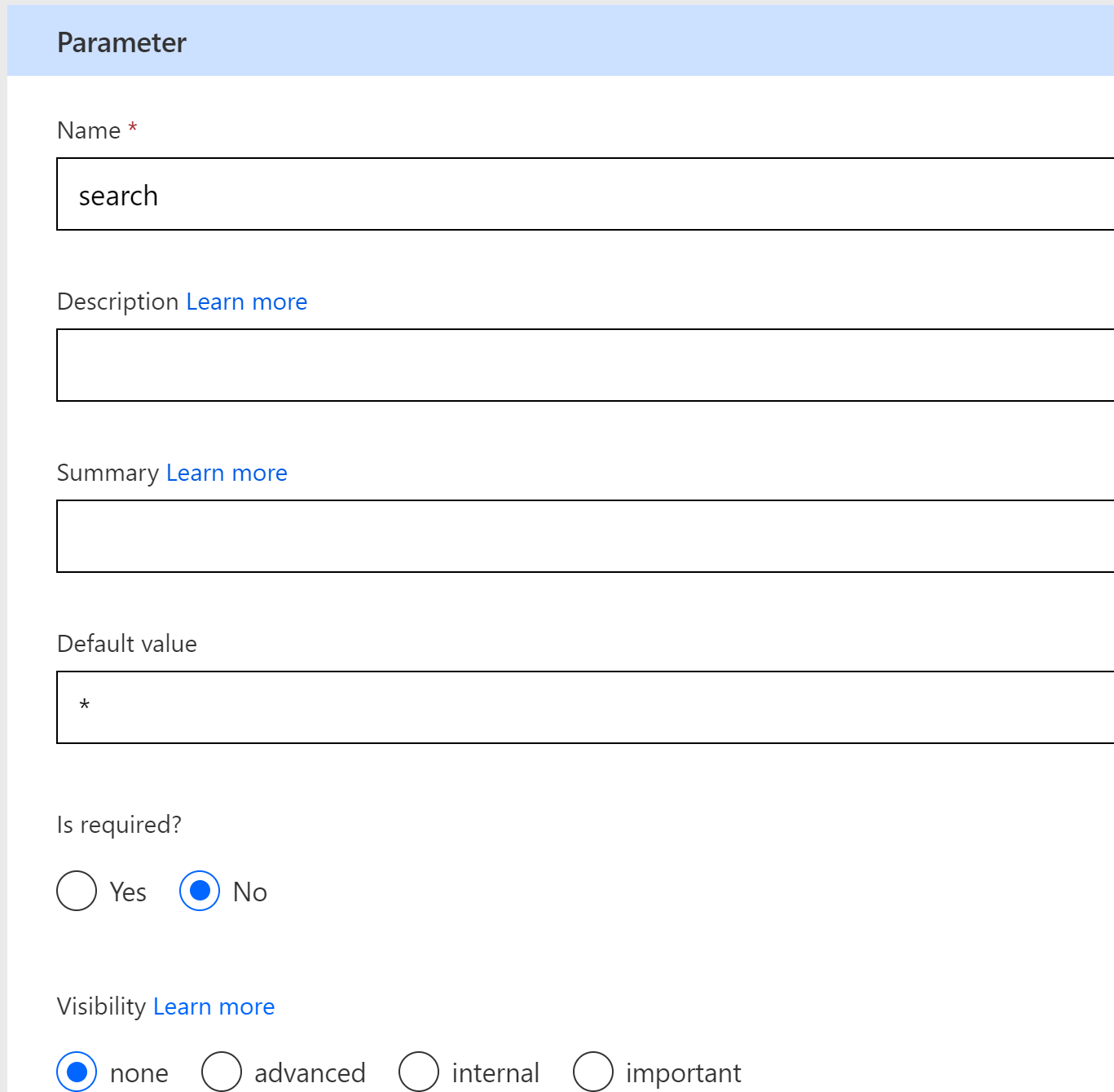


1. Select **Import** to autofill the Request.Connector uses the syntax in the URL to extract parameters from the query: search, select, and api-version parameters become configurable as you progress through the wizard.

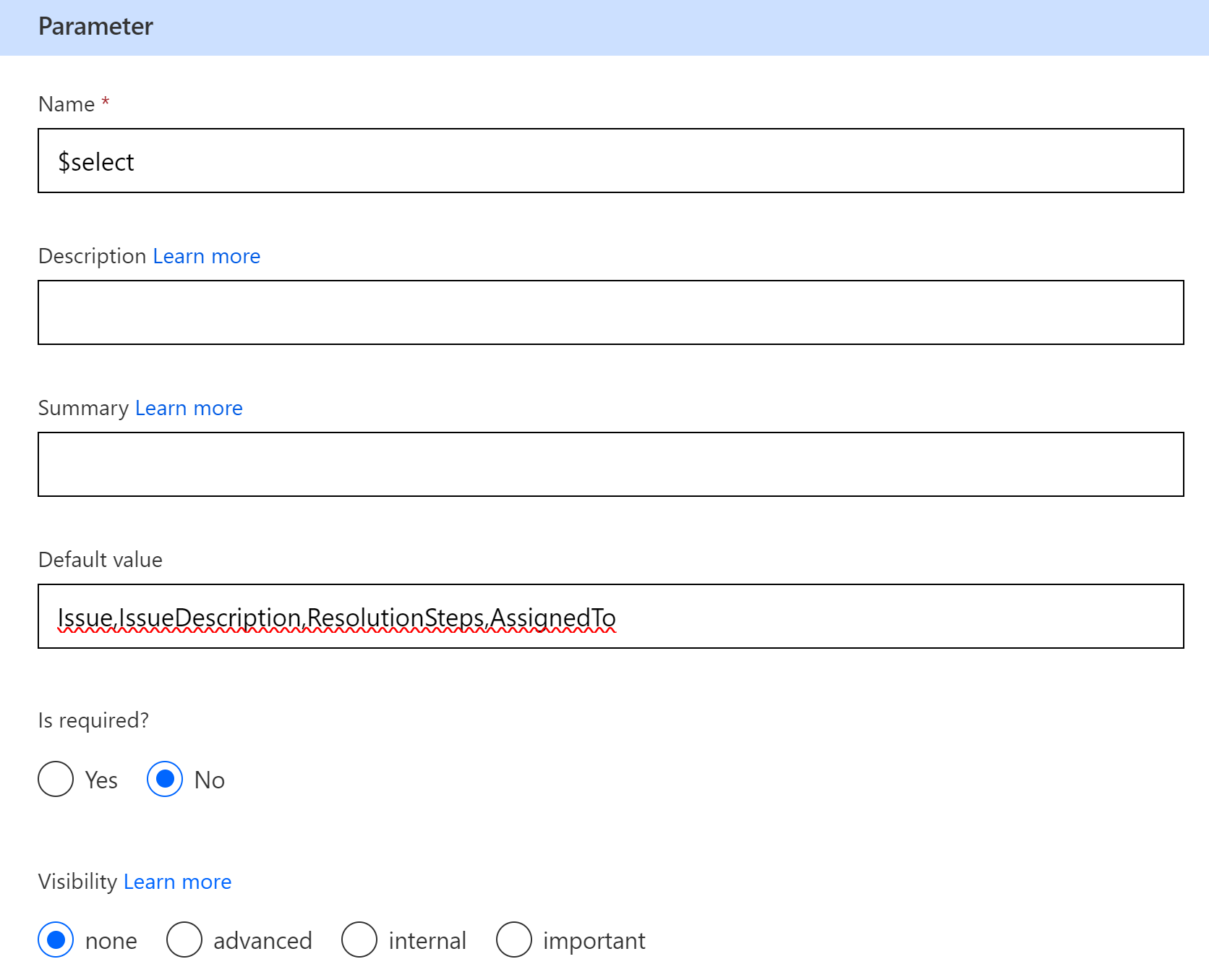


1. Complete setting the parameter metadata by clicking the **...** symbol next to ‘search’ the parameter and select Edit. Set \* as the **default value**, set **required** as *False* and set **visibility** to *none*.

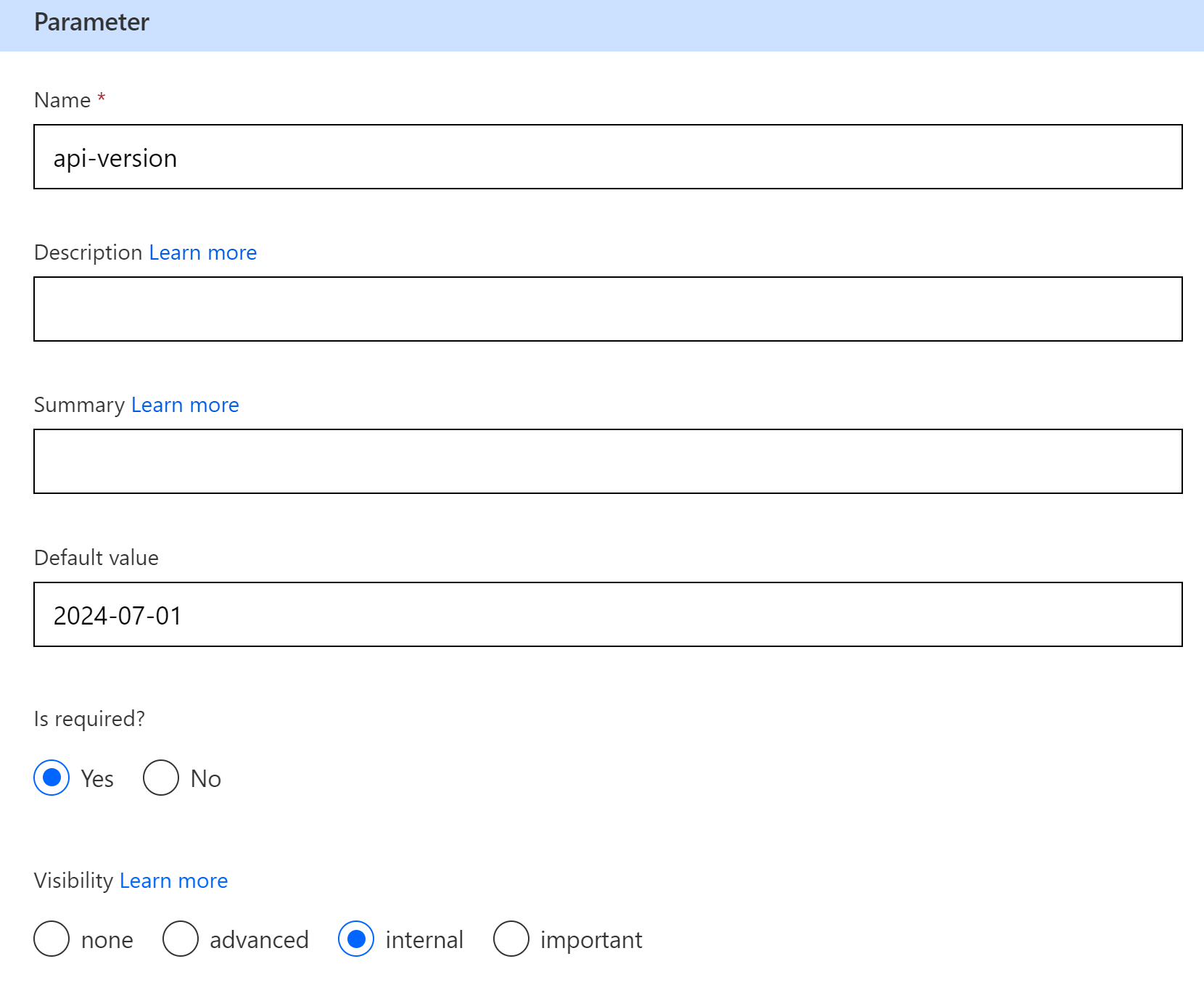
Select **Back** to return to the Request page after each parameter update.



1. For *select*: Set Issue,IssueDescription,ResolutionSteps,AssignedTo as the **default value**, set **required** to *False*, and set **visibility** to *none*.



1. For *api-version*: Set 2024-07-01 as the **default value**, set **required** to *True*, and set **visibility** as *internal*.



1. For *Content-Type*: Set default value to application/json.
2. Scroll down to the Response section. Select **"Add default response"**. This step is critical because it helps Power Apps understand the schema of the response. Power Apps only needs a few results to detect the schema. You can copy the following response into the 'Body’ field and Select **Import**:

*{*

*"@odata.context": "https://ppcc-aippworkshop-svc.search.windows.net/indexes('azuresql-index')/$metadata#docs(\*)",*

*"value": [*

*{*

*"@search.score": 0.75866544,*

*"Issue": "Application Not Responding",*

*"IssueDate": "2024-09-07T00:00:00Z",*

*"IssueDescription": "A specific software application becomes unresponsive and freezes frequently.",*

*"SubmittedBy": "Lucy Park",*

*"ResolutionSteps": "1. Force* *close the application via Task Manager.\n2. Clear app cache and temp files.\n3. Check for software updates.\n4. Reinstall the application if* *issue persists.\n5. Run System File Checker (sfc /scannow) to check for Windows corruption affecting the app.",*

*"AssignedTo": "Laura Wilson",*

*"Status": null,*

*"TicketID": "MTAyMzU2Ng2"*

*},*

*{*

*"@search.score": 0.75061595,*

*"Issue": "Blue Screen of Death (BSOD)",*

*"IssueDate": "2024-09-03T00:00:00Z",*

*"IssueDescription": "PC crashes and displays a blue screen with error codes like \"IRQL\_NOT\_LESS\_OR\_EQUAL.\"",*

*"SubmittedBy": "Alex Brown",*

*"ResolutionSteps": "1. Note down the error code displayed.\n2. Update all drivers via Device Manager.\n3. Run a memory diagnostic tool (search \"mdsched\").\n4. Check for overheating, ensure adequate ventilation.\n5. If recurring, run System File Checker with sfc /scannow and DISM tool for Windows corruption fixes.",*

*"AssignedTo": "Tony Adams",*

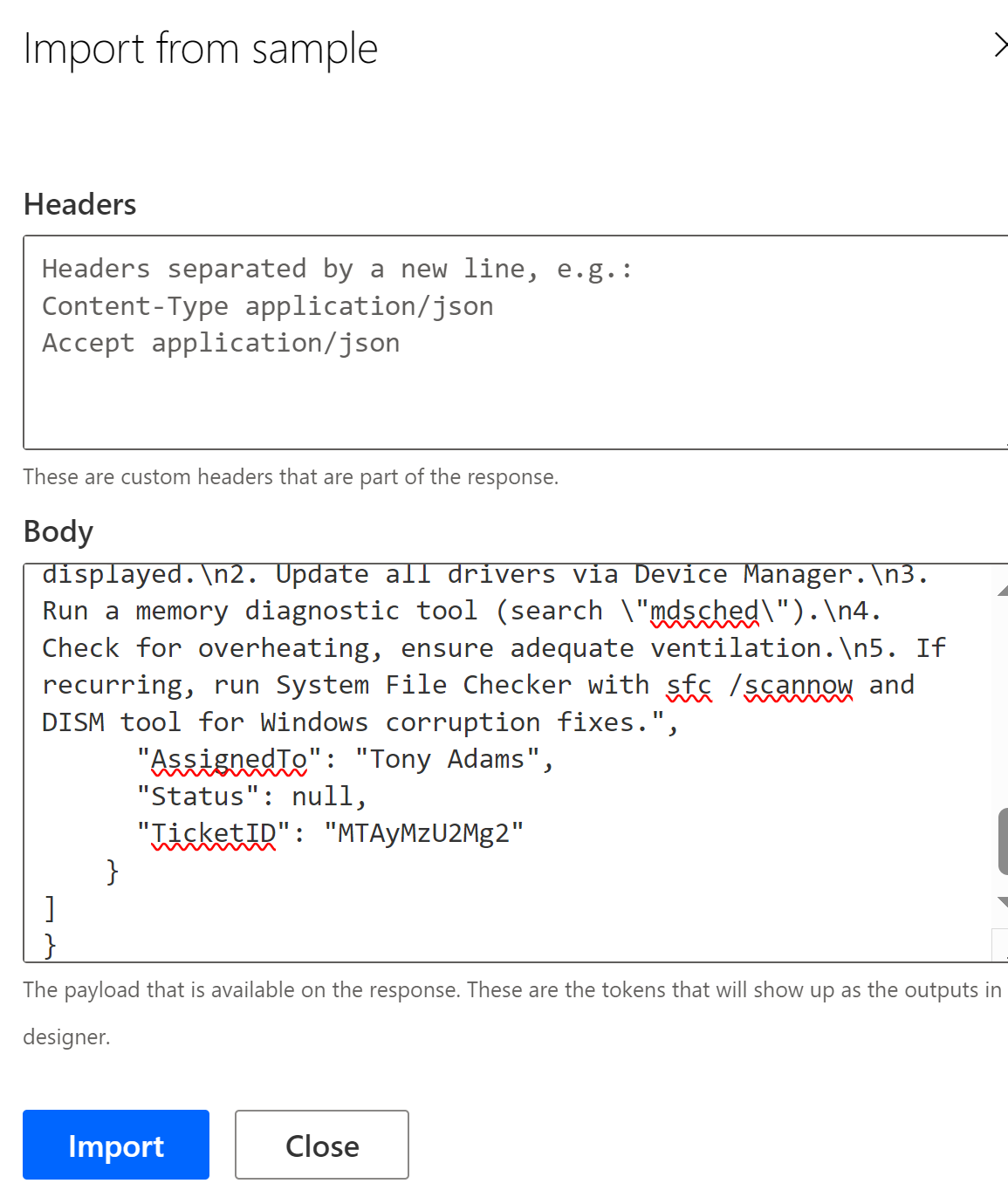
*"Status": null,*

*"TicketID": "MTAyMzU2Mg2"*

*}*

*]*

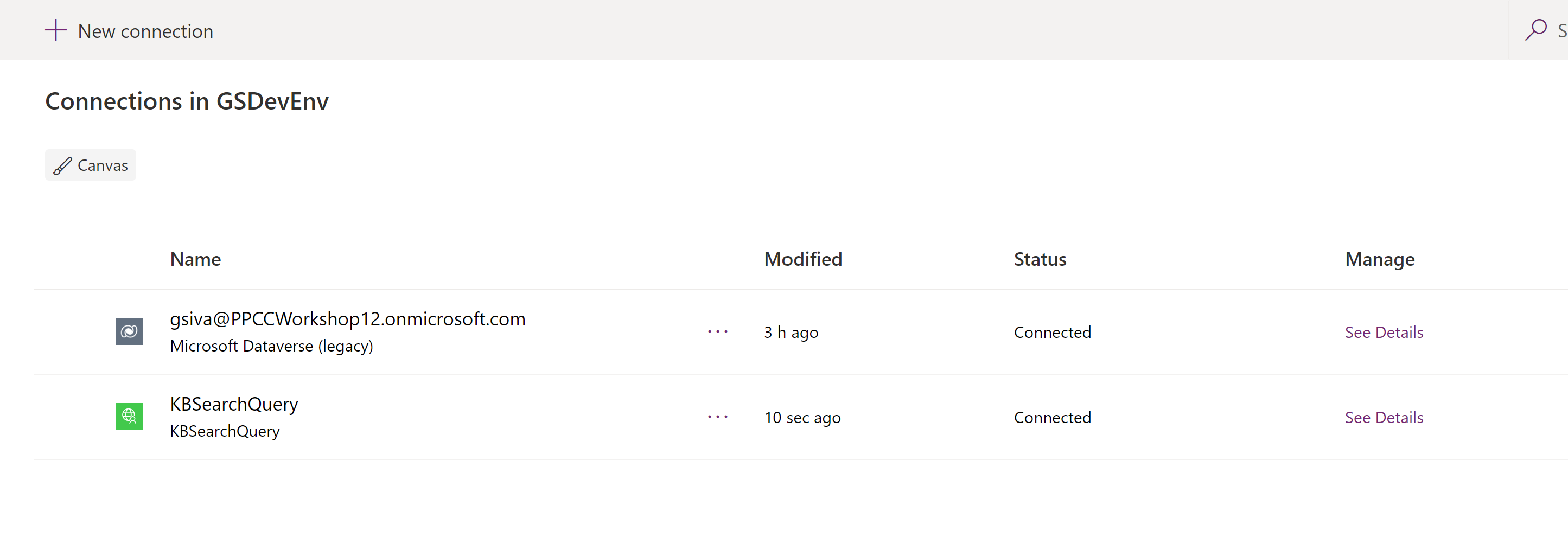
*}*



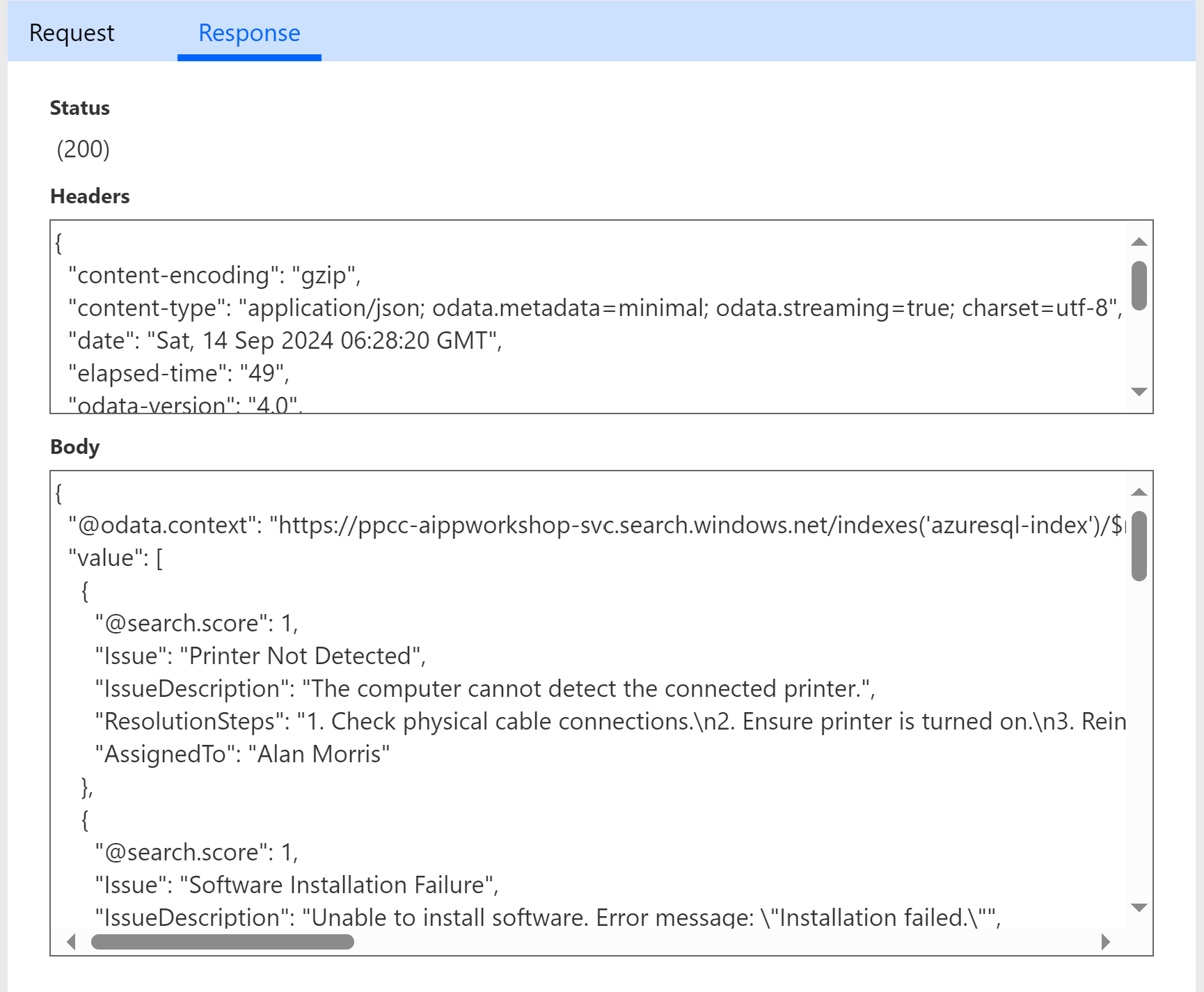
1. Select **Create connector** on the top right to save the definition.
2. In the drop down list of operations, select **6. Test**. In **Test Operation**, select **+ New Connection**. Enter a query API key below and select **Create**. This is an Azure AI Search query for read-only access to an index.

CAjXTKaM6ImkPHeG1QwMa4BUgN0Cl55yMkDIctkPAcAzSeC4MwHo

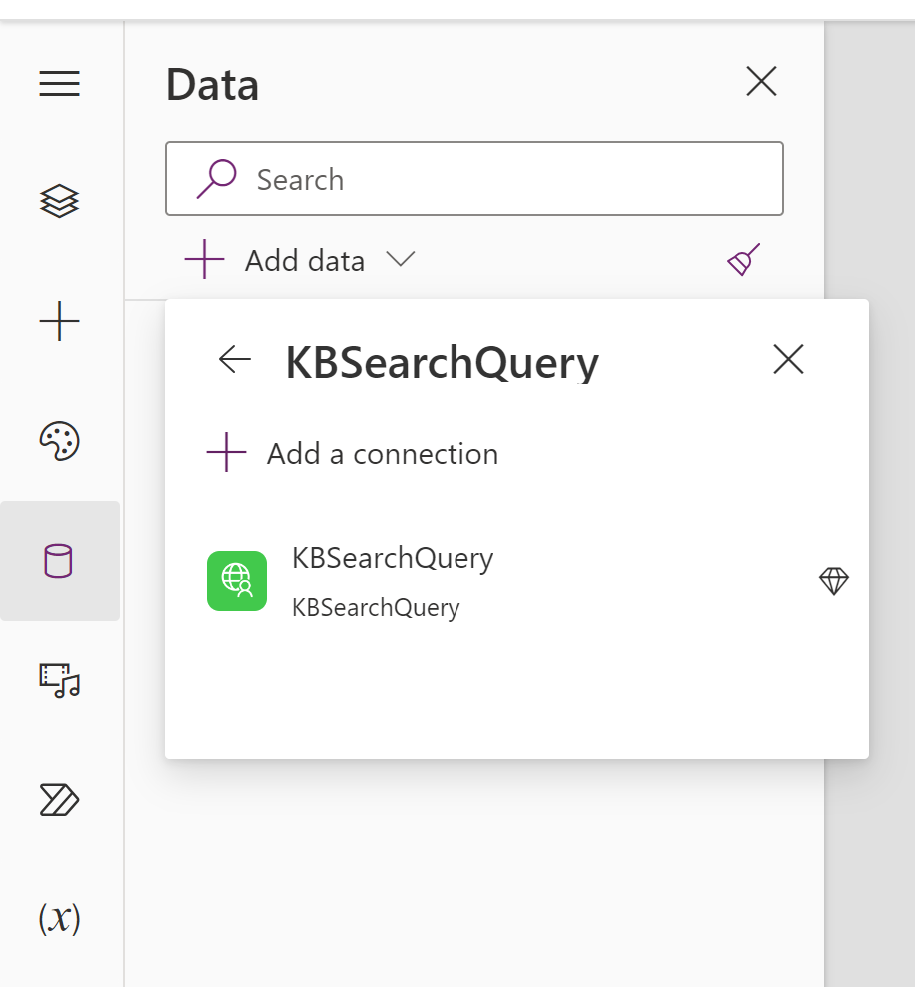
1. This should take you to the connections screen with a new connection created for the custom connector.

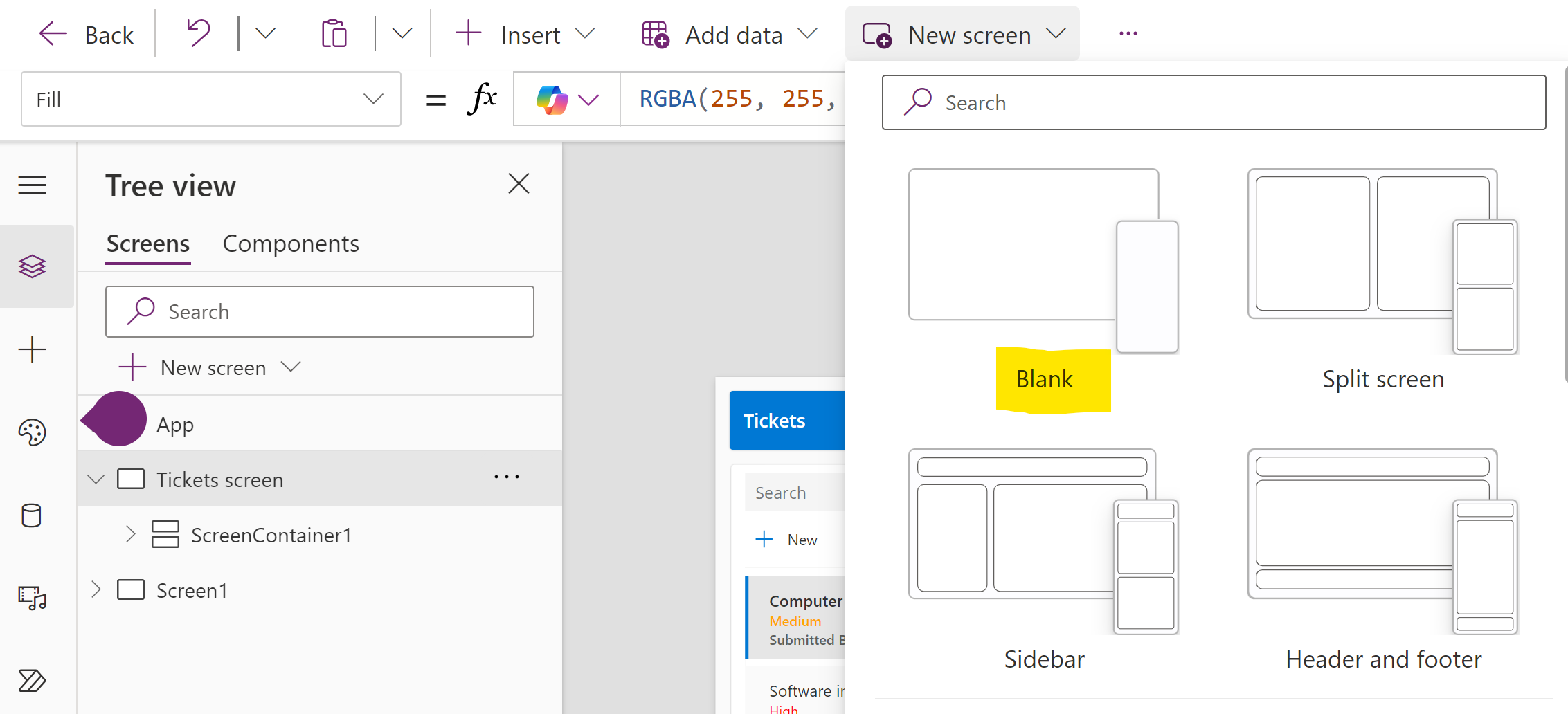


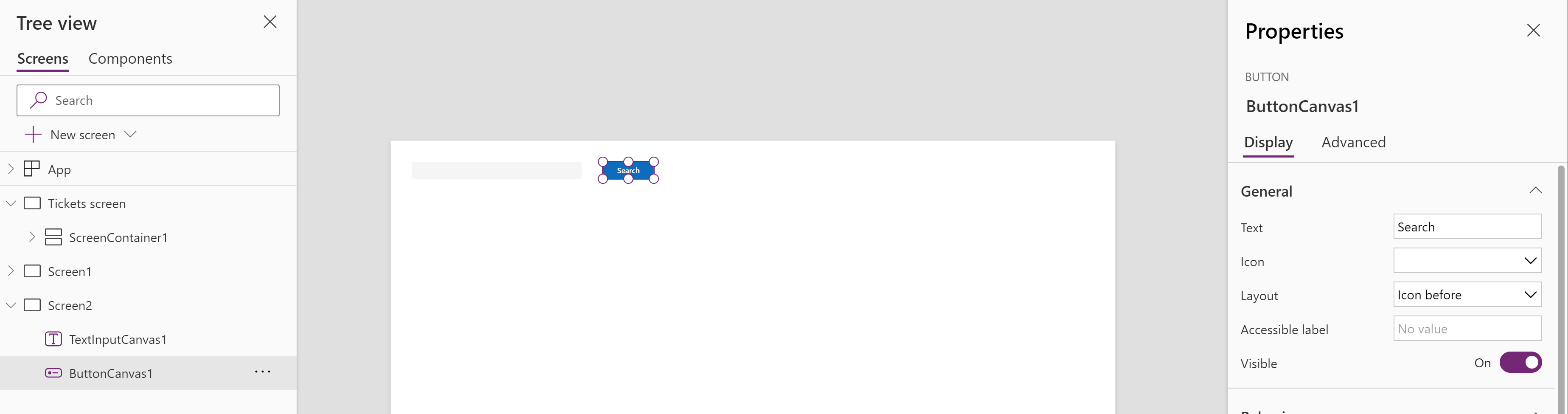
1. Click on Custom Connectors on the left menu, edit the KBSearchQuery connector > In the drop-down list of operations, select **6. Test**. The new connection should appear in the Connections sections. Select the **Test operation** button. If you're successful you should see a 200 status, and in the body of the response you should see JSON that describes the search results.



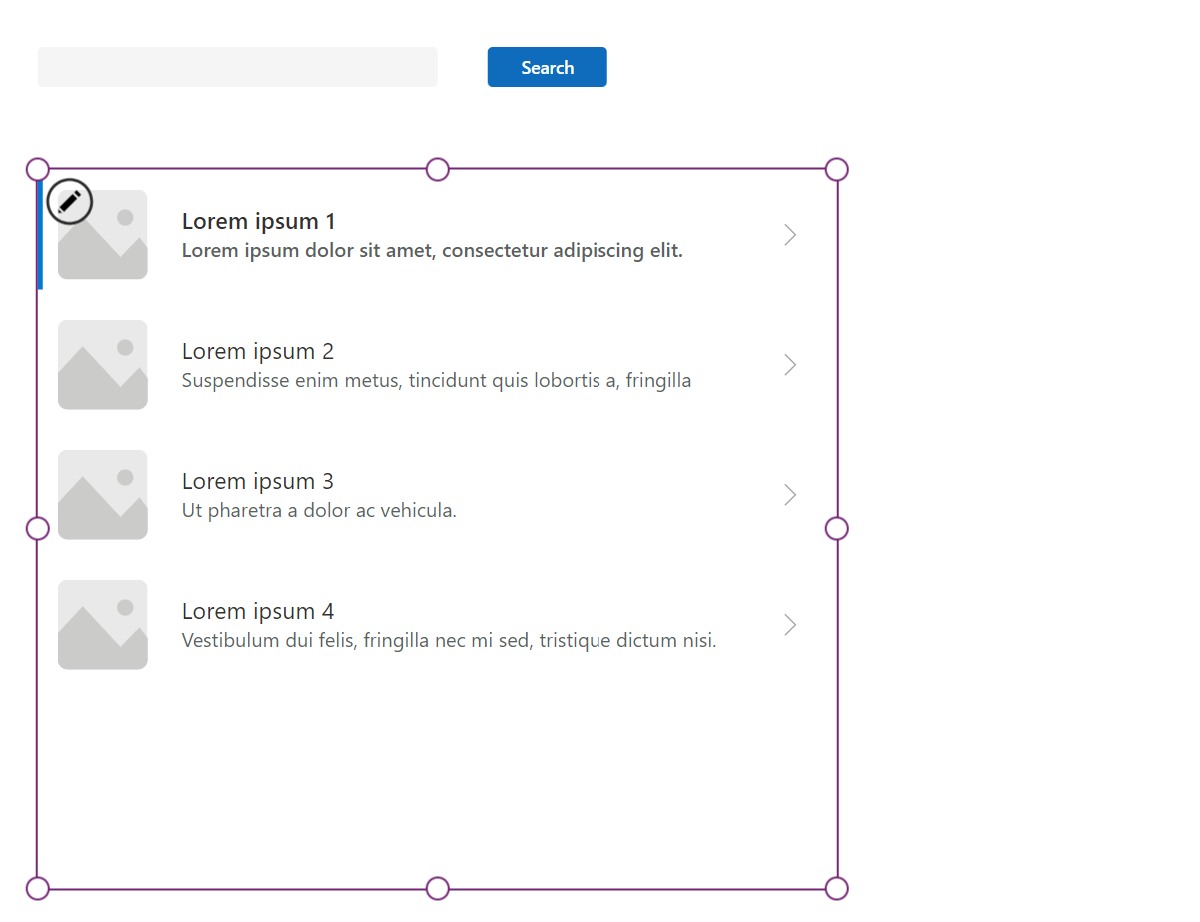
### **Add custom connector to Power App**

1. Navigate to Apps on the left menu and edit the Power App you created as part of Lab 1.
2. Select the **Data** tab, select **Add data**, and then find the new Connector you have just created (Search for KBSearchQuery). Select to **Add a connection**.
3. Insert New Blank Screen > Insert a Text Input control and a Button control. Click on the button control and change ‘Text’ property on the property pane to ‘Search’.





1. Insert a vertical gallery control

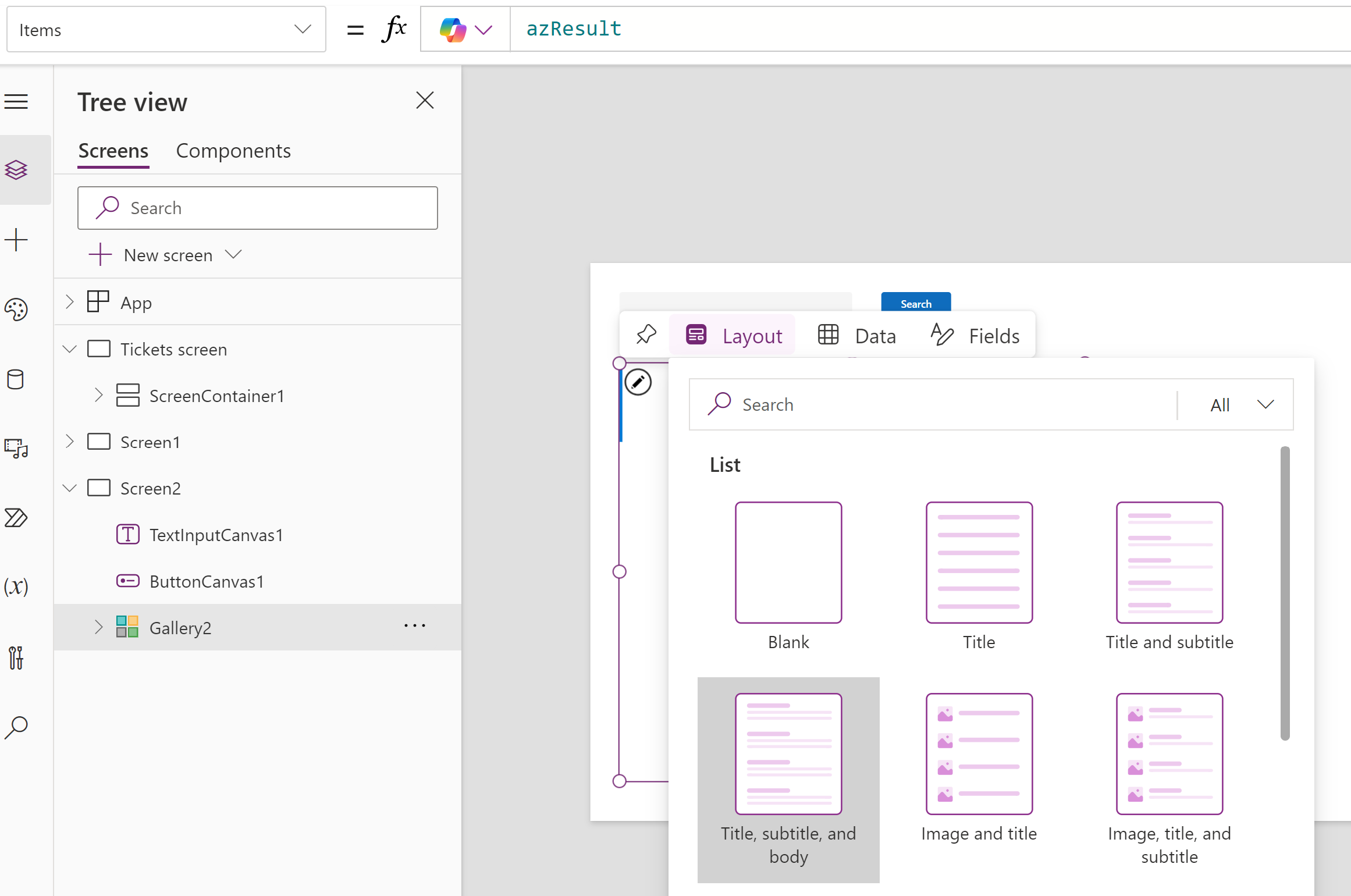


1. To make the **Search button** issue a query, paste the following action into **OnSelect.** This action causes the button to update a new collection called *azResult* with the result of the search query, using the text in the *TextInputCanvas1* text box as the query term.

*If(!IsBlank(TextInputCanvas1.Value), ClearCollect(azResult,*

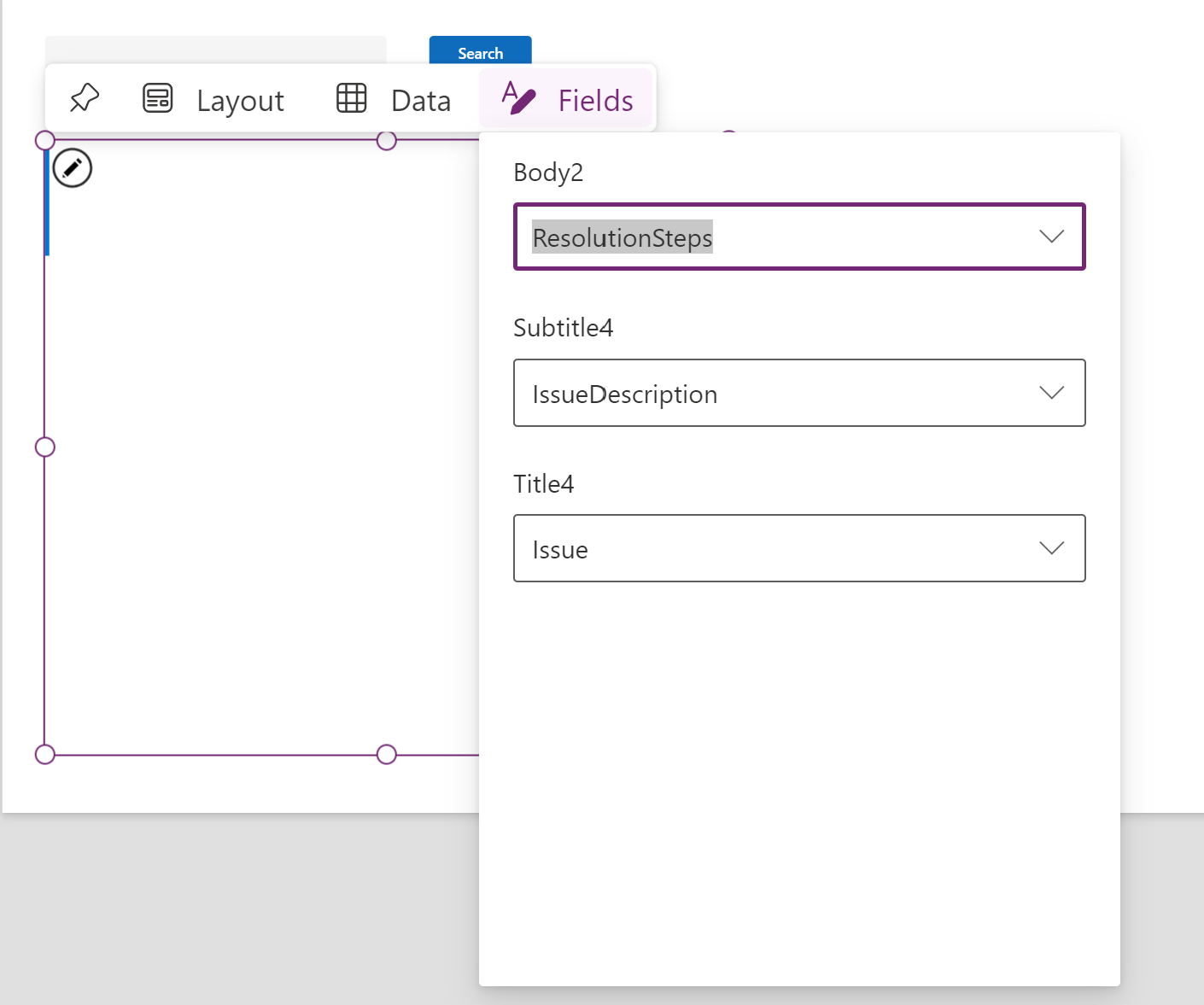
*KBSearchQuery.Query({search: TextInputCanvas1.Value}).value))*

1. Link the Vertical Gallery control to the *azResult* collection that was created when you completed the previous step. Select the gallery control and perform the following actions in the properties pane.
   1. Set **Items** to *azResult*.
   2. Select a **Layout** that works for you based on the type of data in your index. In this case, we used the *Title, subtitle and body* layout.



* 1. **Edit Fields and** select the fields you would like to visualize as seen below.

Since we provided a sample result when we defined the connector, the app is aware of the fields available in your index.



1. Press **F5** to preview the app. Type ‘No Internet Connection’ in the input text box and click **Search.** Power App passes the search terms to the custom connector we built which in turn queries AI search and returns most relevant items from Knowledge Base. The text could appear to be cut off. Adjustments to the controls can enable all the content to fit within the screen
2. Set the following values for the Gallery to create space for KB content.
   1. Set **Width** property on the Gallery = 1286 & height =600
   2. Set **Template** property on the Gallery = 200
   3. Set **Auto Height** property on Body2 control = true
   4. Move the Title, Subtitle and Body field upwards as needed to have enough space
3. Publish the app to commit changes.

