

Power Automate Design Documentation for Email Notification System

Actionable Message

Overview

The actionable message is a technology that allows us to embed interactive elements like buttons (thumbs-up/thumbs-down), comments, etc., into our outlook emails without needing to open a new tab to visit a website. Figure 1 below shows an example of a poll in Outlook email. Example: Poll in Outlook email

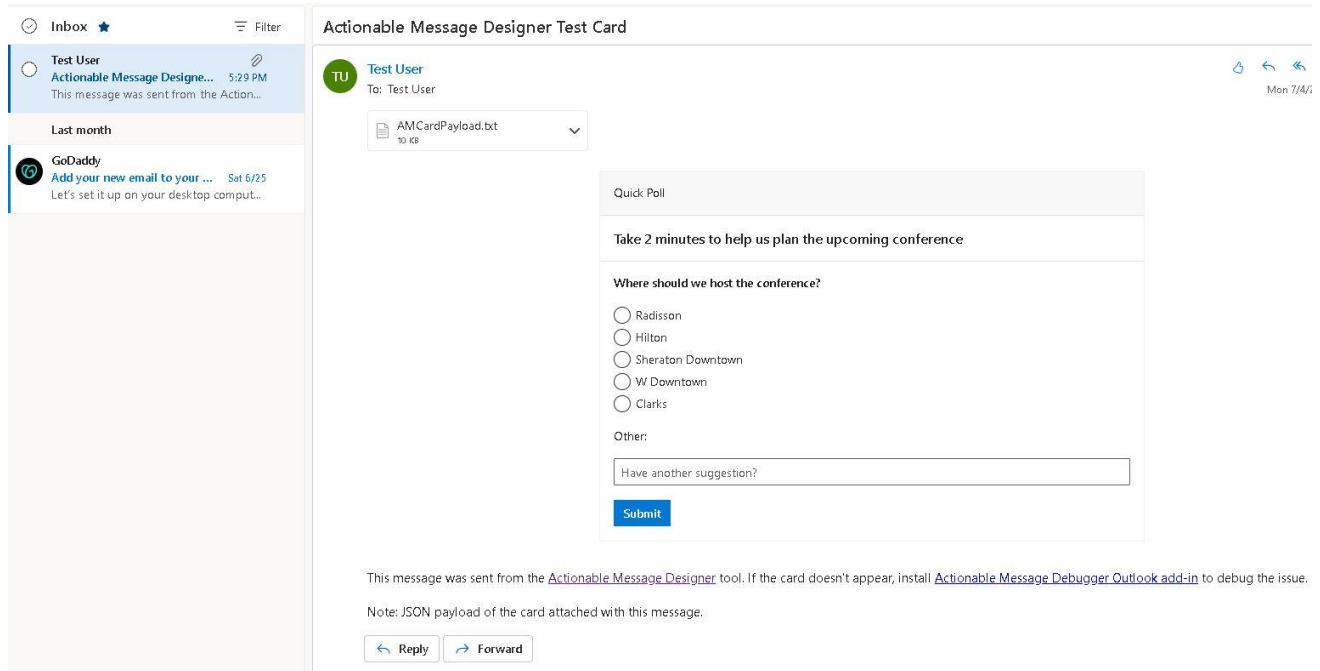


Figure 1: A poll in Outlook email

Power Automate Design Steps

Step 1: Use the JSON code to build the Actionable Message card based on the given Mockup

Explain: Outlook Actionable Messages cards are designed using the Adaptive Card format. And the Adaptive Card is designed in JSON code.

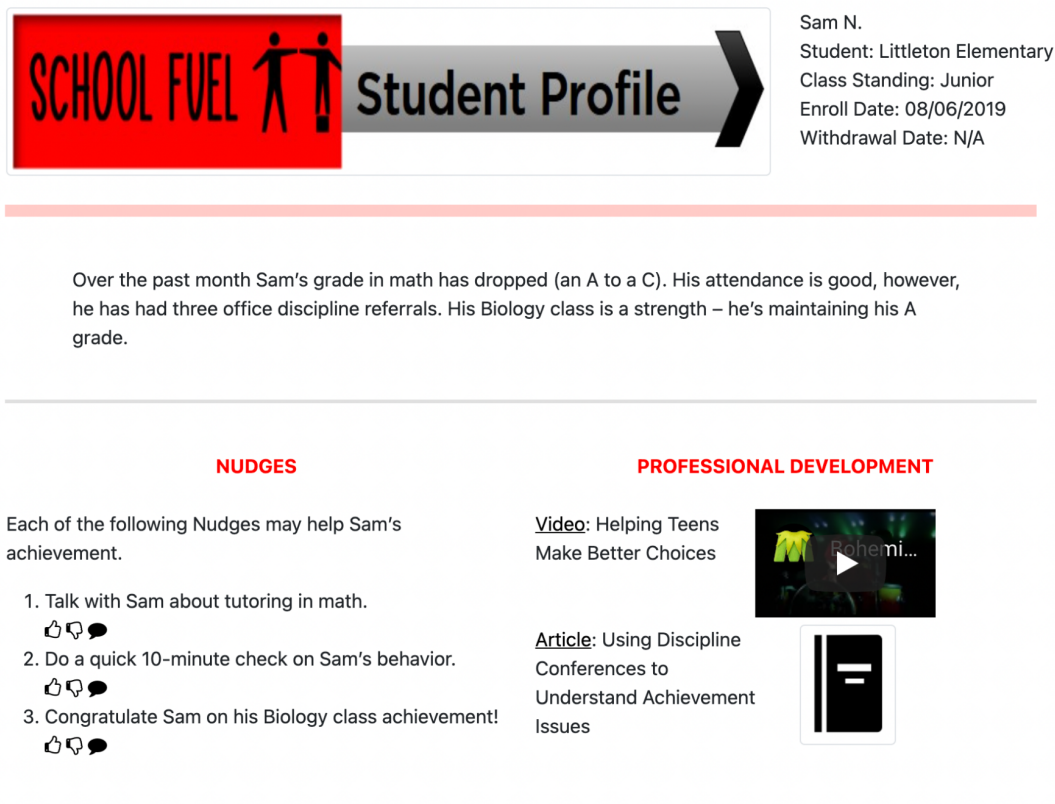


Figure 1.1: Mockup

Tool: [Actionable Message Designer](#)

We can use the **Actionable Message Designer** to create the mockup(Figure1.1). Actionable Message Designer (Figure 1.2) is a tool that provides a drag-and-drop experience to build and tweak adaptive cards quickly. It can also generate JSON code (Figure 1.3).

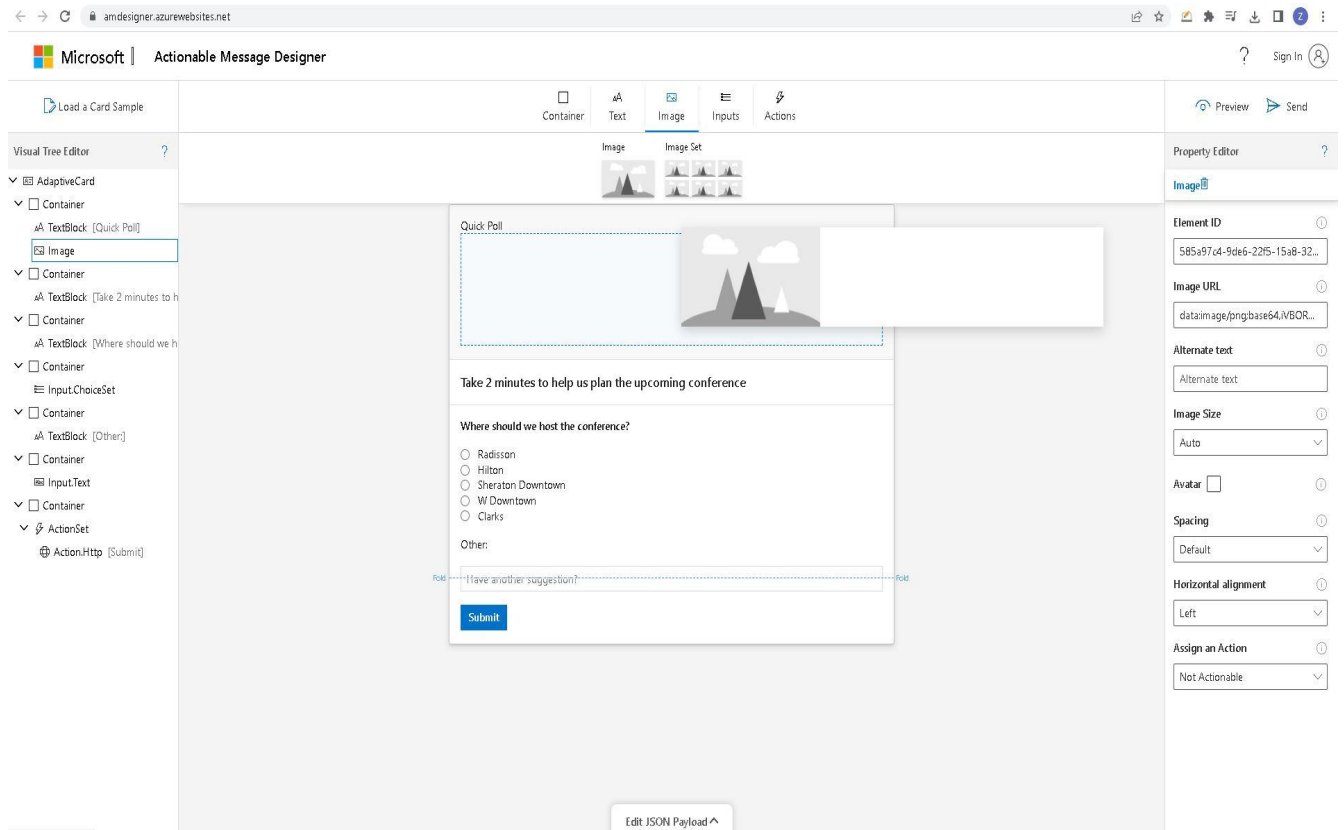


Figure 1.2: Actionable Message Designer

Microsoft | Actionable Message Designer

Visual Tree Editor

- AdaptiveCard
 - Container
 - TextBlock [Quick Poll]
 - Container
 - TextBlock [Take 2 minutes to h]
 - Container
 - TextBlock [Where should we h]
 - Container
 - Input.ChoiceSet
 - Container
 - TextBlock [Other]
 - Container
 - Input.Text
 - Container
 - ActionSet
 - Action.Http [Submit]

Quick Poll

Take 2 minutes to help us plan the upcoming conference

Where should we host the conference?

- ☐ Radisson
- ☐ Hilton
- ☐ Sheraton Downtown
- ☐ W Downtown
- ☐ Clarks

Other:

Have another suggestion?

Edit JSON Payload

```

1 {
2   "type": "AdaptiveCard",
3   "body": [
4     {
5       "type": "Container",
6       "style": "emphasis",
7       "items": [
8         {
9           "type": "TextBlock",
10          "text": "Quick Poll",
11          "wrap": true
12        },
13      ],
14      "padding": "Default"
15    },
16    {
17      "type": "Container",
18      "id": "d9d9d9d9-2a13-7f7f-4608-438a9f7777f7"
19    }
20  ]
21 }

```

Select an element to customize its properties

Figure 1.3: JSON code

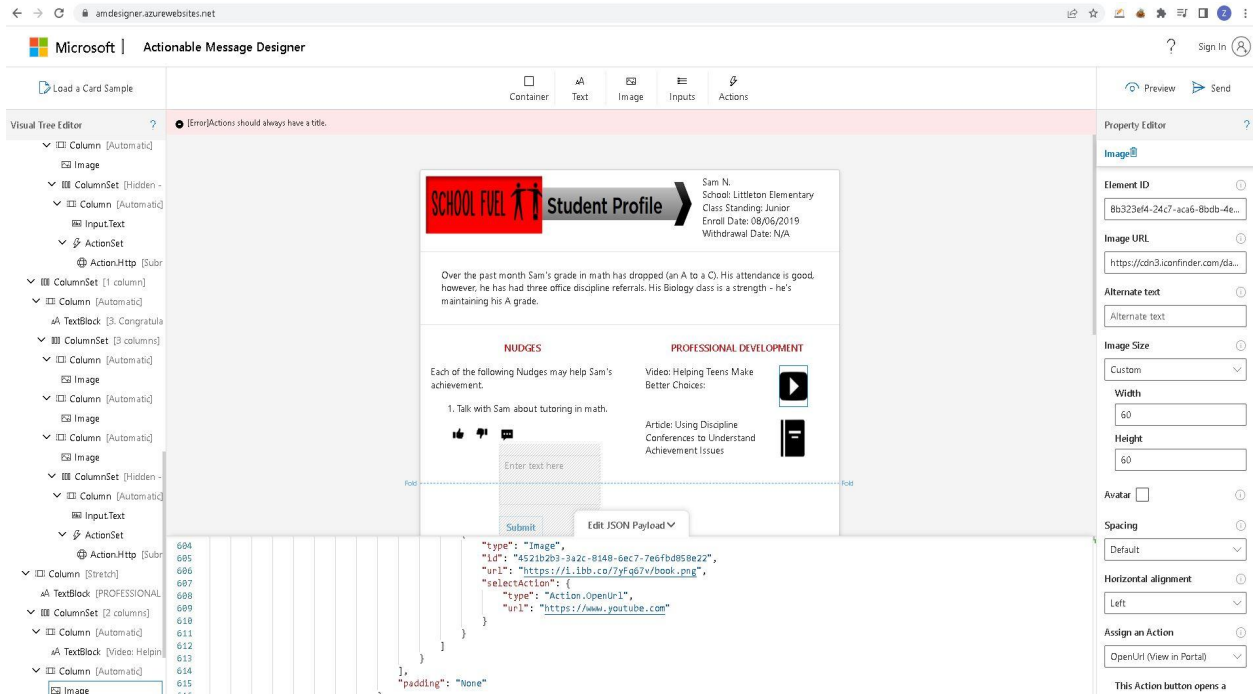


Figure 1.3: Using Actionable Message Designer to build mockup
(You can download the JSON code in Appendix1.)

Step 2: Using Power Automate Flow to Receive Response from Actionable Message

We have some feedback buttons (thumbs up, thumbs down, comment) on our nudges part (Figure 2.1). So, we need a server or endpoint to receive these feedback responses.

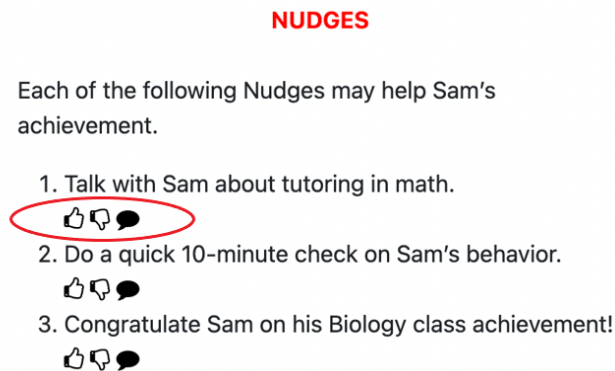
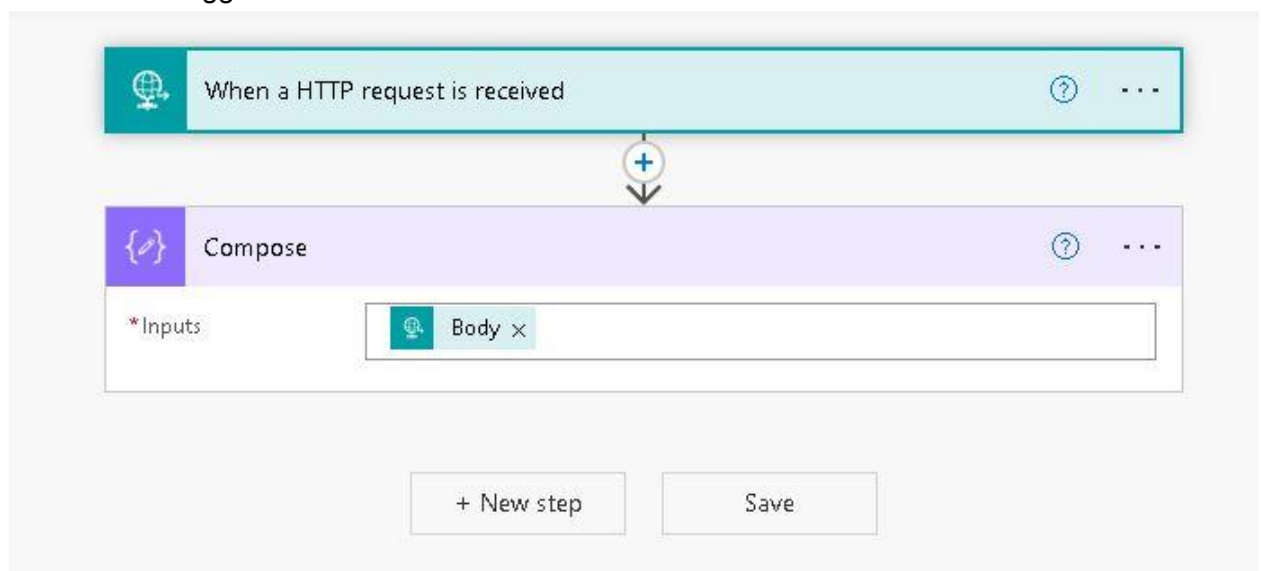


Figure 2.1: Feedback Buttons

For the server or endpoint, we can use Power Automate flow. Power Automate can make and receive HTTP requests. HTTP requests can interact with REST APIs, of virtually any software or application.

- In Power Automate, we can use “HTTP request” as a trigger, and “Compose” with the input “Body” as an action. The reason we added that compose with the body is every flow needs a trigger and an action to be able to save.



- b. Then we can save the flow. The endpoint URL will be generated after saving. We can just copy it. (This URL will be used for feedback buttons. When the client clicks these buttons, the button will send an HTTP request to the Power Automate server)

The screenshot displays the Power Automate flow editor interface. The first step, 'When a HTTP request is received', is highlighted with a light blue header. It includes a 'Copy Url' button in the top right corner. Below the header, the 'HTTP POST URL' field contains the endpoint: `https://prod-32.westus.logic.azure.com:443/workflows/fb723784d65743...`. The 'Request Body JSON Schema' section is currently empty, with a link to 'Use sample payload to generate schema' and a 'Show advanced options' dropdown. A connector arrow points down to the second step, 'Compose', which has a light purple header. The 'Compose' step's 'Inputs' section shows a single input named 'Body'. At the bottom of the editor, there are two buttons: '+ New step' and 'Save'.

- c. Back to the Actionable Message Designer, we can select one of the buttons and insert the endpoint URL.

The screenshot displays the 'Actionable Message Designer' interface. On the left, a 'Student Profile' card for 'Sam N.' from 'Littleton Elementary' is shown. Below it, a 'NUDGES' section contains three numbered items. The first nudge, 'Talk with Sam about tutoring in math.', has a button icon circled in red. To the right, a 'PROFESSIONAL DEVELOPMENT' section shows video and article thumbnails. On the far right, the 'Property Editor' panel is open, showing the 'Assign an Action' dropdown set to 'Http (Submit)'. Below this, the 'Url' field is circled in red and contains the text 'https://prod-32.westus.logic.a...'. A tooltip above the field reads 'Insert the URL/server address where this request will be made.' The 'Body' field is empty, and a note states 'This action has no header.' with an 'Add a new header' button below it.

- d. For the body of the button, we can insert the data that needs to be sent to the Power Automate server. The format of body: `{"Anyname": "{{elementID.value}}"}"`

This close-up shows the configuration for the button's action. The 'Url' field contains the endpoint 'https://prod-32.westus.logic.az...'. The 'Body' field contains a JSON payload: `{"textContent1": "{{inputComment1.value}}"}"`.

- e. For the header, we must use the following information.

The screenshot shows an API client interface with the following fields:

- Url**: `https://prod-32.westus.logic.azure...`
- Body**: `{"textContent1": "{{inputComment1.value}}"}"`
- Key**: `Authorization`
- Value**: `Enter header value`
- Key**: `Content-Type`
- Value**: `application/json`

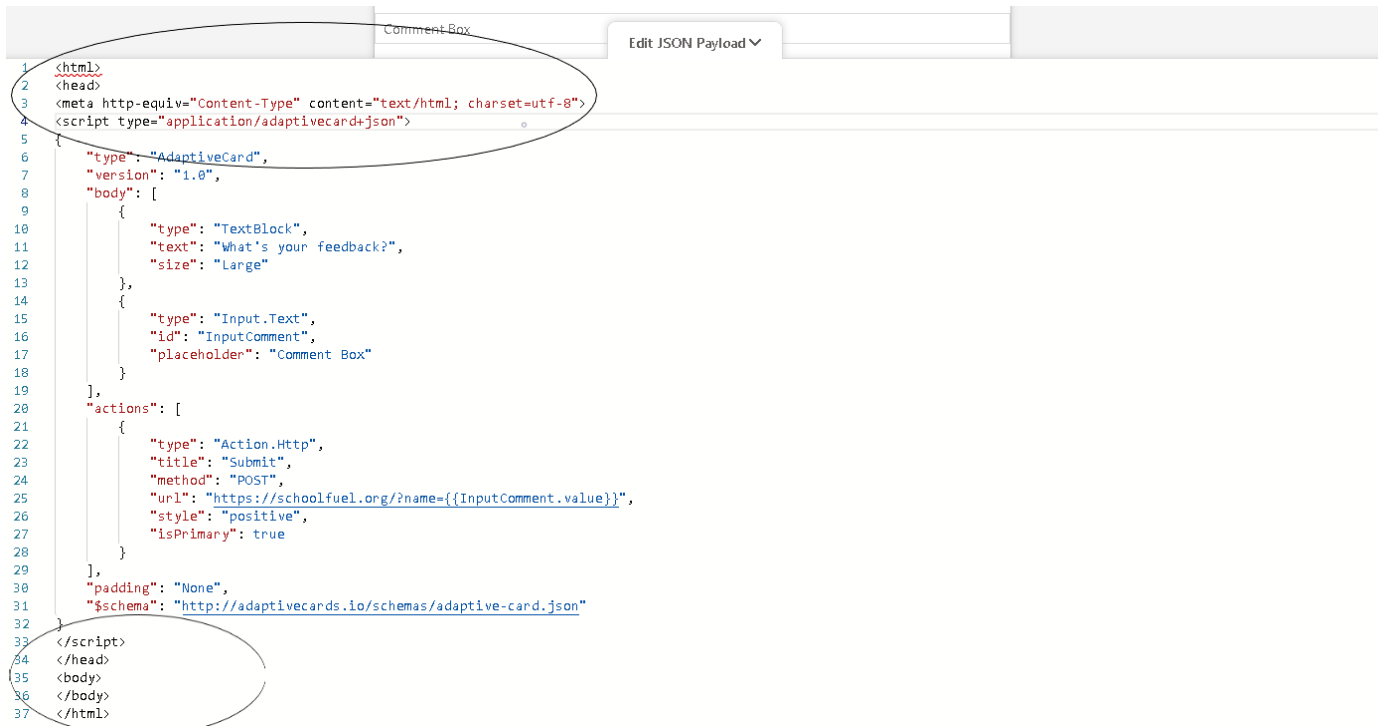
A red oval highlights the 'Authorization' key and the 'application/json' value. A blue box highlights the 'application/json' value.

Add a new header

- f. Repeat steps c, d, and e for the rest of the buttons.
g. Save the output JSON code

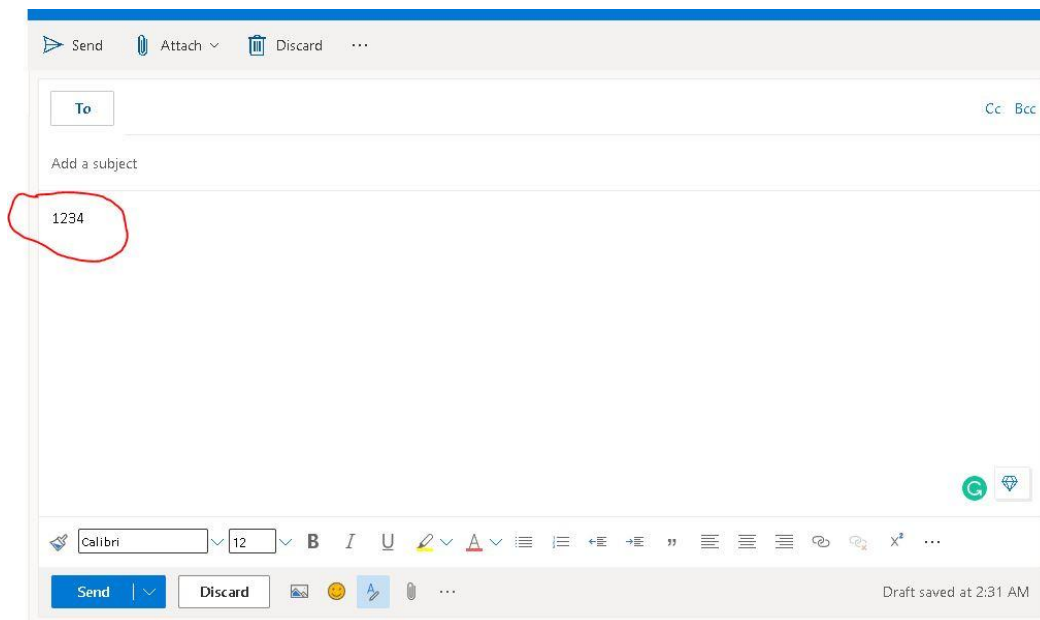
Step 3: Test/Send the built actionable message card via Outlook email

a. To embed an actionable message card in an email message, we need to wrap the card in a <script> tag. The <script> tag is then inserted into the <head> of the email's HTML Body.

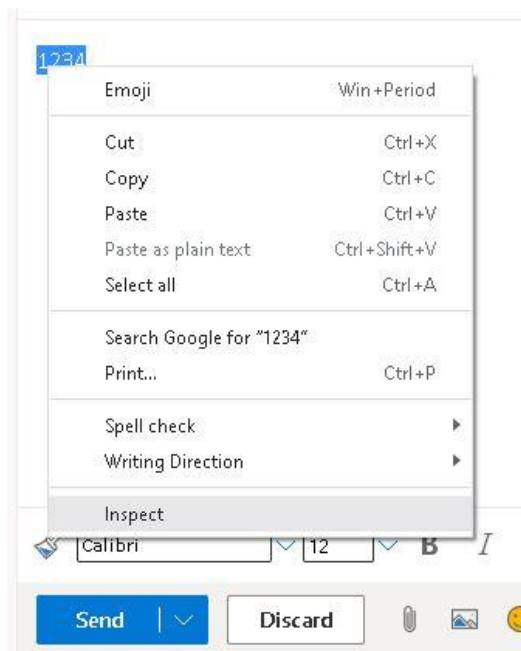


```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
4 <script type="application/adaptivecard+json">
5 {
6   "type": "AdaptiveCard",
7   "version": "1.0",
8   "body": [
9     {
10      "type": "TextBlock",
11      "text": "What's your feedback?",
12      "size": "Large"
13    },
14    {
15      "type": "Input.Text",
16      "id": "InputComment",
17      "placeholder": "Comment Box"
18    }
19  ],
20  "actions": [
21    {
22      "type": "Action.Http",
23      "title": "Submit",
24      "method": "POST",
25      "url": "https://schoolfuel.org/?name={{InputComment.value}}",
26      "style": "positive",
27      "isPrimary": true
28    }
29  ],
30  "padding": "None",
31  "$schema": "http://adaptivecards.io/schemas/adaptive-card.json"
32 }
33 </script>
34 </head>
35 <body>
36 </body>
37 </html>
```

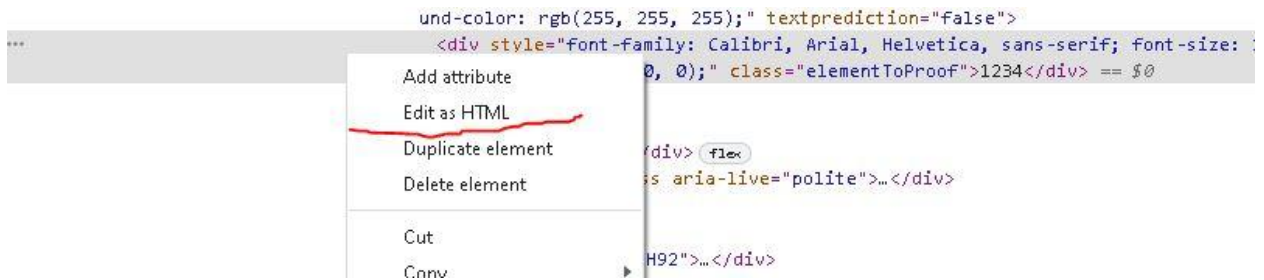
- b. We just copy this entire bit of code. Then we go to the outlook email. Here we want to add placement “1234” to the body of the email. The placement can be anything.



- c. Highlight the placement, then right-click and inspect.



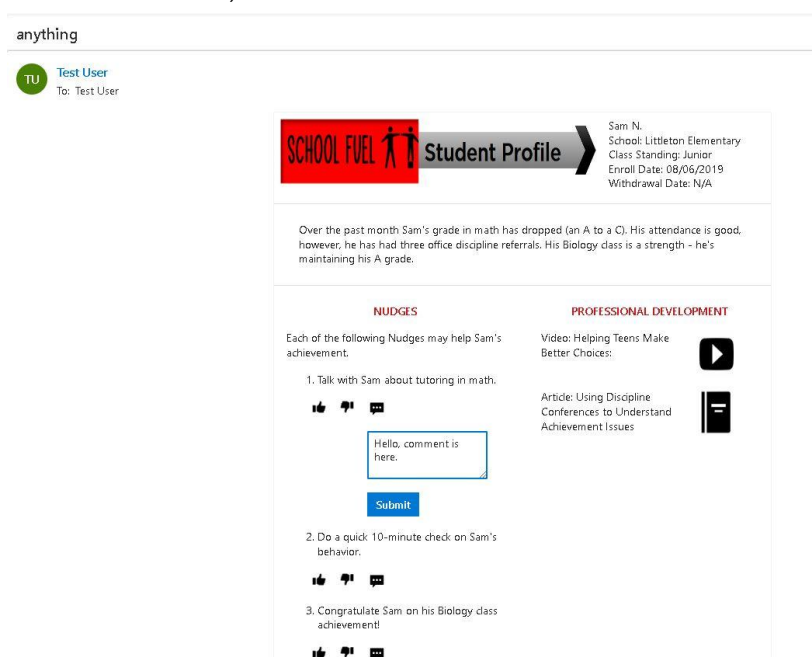
- d. Navigate up to the highlight section, then right-click and choose “Edit as HTML”



- e. Navigate down to the placement “1234”, and replace it with the code from the JSON payload. And click out of that.



- f. Send the email to the same email address. For example testuser@schoolfuel.org to testuser@schoolfuel.org
- g. We will receive the actionable message in the outlook email. We can type something in the comment box, and submit it.



h. Back to the first flow, we can see the response is received.

28-day run history ⓘ [All runs](#)

Start	Duration	Status
Aug 29, 12:52 AM (11 sec ago)	37 ms	Succeeded

🟢 Your flow ran successfully.

```
graph TD; Trigger[When a HTTP request is received] --> Compose[Compose];
```

Compose 0s

INPUTS [Show raw inputs](#)

Inputs

```
{
  "textContent1": "Hello, comment is here."
}
```

OUTPUTS [Show raw outputs](#)

Outputs

```
{
  "textContent1": "Hello, comment is here."
}
```

Step 4: How to Download Output Data from Power Automate

- a. Select the My flows link, and then select the flow for which you want to export the run history. In the 28-day run history, select all runs.

The screenshot shows the Power Automate interface. On the left is a navigation pane with 'My flows' selected. The main area displays the 'AC handler' flow details. Below the details is the '28-day run history' table. The 'All runs' button is highlighted with a yellow circle.

Start	Duration	teacherComm...	Status
Nov 11, 12:47 AM (1 h ago)	48 ms	"12333"	Succeeded
Nov 11, 12:40 AM (1 h ago)	27 ms		Succeeded
Nov 11, 12:38 AM (1 h ago)	02 ms	"test1"	Succeeded
Nov 10, 02:23 PM (11 h ago)	21 ms		Succeeded
Nov 10, 02:17 PM (11 h ago)	46 ms	"123456"	Succeeded
Nov 10, 02:12 PM (11 h ago)	30 ms		Succeeded
Nov 10, 02:04 PM (11 h ago)	49 ms	"[[studentInfo1.value]]"	Succeeded

- b. Choose "Get.csv file" and you will be able to download the data.

The screenshot shows the Power Automate interface with the 'AC handler' flow selected. The 'Run history' table is displayed. The 'Get.csv file' button is highlighted with a yellow circle.

Start time	Duration	teacherComment1	Status
Nov 11, 12:47 AM (1 h ago)	48 ms	"12333"	Succeeded
Nov 11, 12:40 AM (1 h ago)	27 ms		Succeeded
Nov 11, 12:38 AM (1 h ago)	02 ms	"test1"	Succeeded
Nov 10, 02:23 PM (11 h ago)	21 ms		Succeeded
Nov 10, 02:17 PM (11 h ago)	46 ms	"123456"	Succeeded
Nov 10, 02:12 PM (11 h ago)	30 ms		Succeeded
Nov 10, 02:04 PM (11 h ago)	49 ms	"[[studentInfo1.value]]"	Succeeded
Nov 10, 02:04 PM (11 h ago)	57 ms	"11111"	Succeeded
Nov 10, 01:17 PM (12 h ago)	18 ms	"hello123"	Succeeded
Nov 10, 11:39 AM (14 h ago)	26 ms	"1"	Succeeded
Nov 10, 01:29 AM (1 d ago)	53 ms	"hello chen"	Succeeded

- c. If we want, we can also add a few more steps to this flow. Those steps allow us to add output data to an excel file, and this excel file can be in any format we want. Here is a tutorial video from Youtube. (link: <https://youtu.be/kbYGFlo0WGC>)

Step 5: How to Share the Power Automate Flow

To share the flow, there are two things we can do.

1. Add an owner to a cloud flow. Adding another owner allows others to edit, update and delete this flow. All owners can also access the run history and add or remove other owners.
2. Share a copy of a cloud flow. So people in our organization will be able to create a copy of this flow.

The screenshot displays the Power Automate web interface. The top navigation bar is blue with the 'Power Automate' logo and a search bar. Below the navigation bar, a toolbar contains icons for 'Edit', 'Share', 'Save As', 'Delete', 'Send a copy', 'Submit as template', 'Export', 'Process insights (preview)', 'Analytics', 'Turn off', and 'Re'. The 'Share' and 'Send a copy' icons are highlighted with red circles. The main content area shows the details of a flow named 'AC handler'. The details panel includes fields for 'Flow' (AC handler), 'Status' (On), 'Owner' (Zezhao Chen), 'Created' (Jul 9, 06:42 PM), 'Modified' (Jul 9, 07:00 PM), 'Type' (Automated), and 'Plan' (This flow runs on owner's plan). Below the details panel, there is a section for '28-day run history' with a table showing columns for 'Start', 'Duration', 'teacherComm...', and 'Status'. The 'All runs' link is visible next to the '28-day run history' header.

Start	Duration	teacherComm...	Status
-------	----------	----------------	--------

Note:

- General Data Protection Regulation (GDPR) requires Power Automate to keep run logs for no longer than 28 days. To maintain a longer history we'll need to manually capture run histories before they are deleted.
- If we want to send the actionable message to other organizations, it actually requires approval from Microsoft. We will be submitting certain details of School Fuel to Microsoft, which after being reviewed and approved, will enable actionable messages for School Fuel's service. For more information, please go to Actionable Message documentation ([link](#)).

Reference:

- Actionable Message documentation ([link](#))
- Actionable Messages step-by-step ([link](#))

Appendix1: [Format 3 JSON code](#)

Email: zchen194@asu.edu If you have any questions about this documentation, feel free to send an email to me.