**Microsoft Adaptive Card POC Plan**

**Goal:** Demonstrate the capability of Microsoft Adaptive Cards to send actionable emails, make HTTP calls with input fields as payload, and address security concerns.

**Phase 1: Foundation & Simple Actionable Email**

**Outcome:** Successfully send a basic actionable email with a simple action.

**Steps:**

1. **Understand Adaptive Cards Basics:**
   * Familiarize yourself with the Adaptive Card schema and JSON structure.
   * Use the [Adaptive Card Designer](https://adaptivecards.io/designer/) to build simple card layouts. Experiment with TextBlock, Input.Text, Action.Submit, and Action.OpenUrl.
   * Pay attention to the "Host app" selection in the designer (especially "Outlook Actionable Messages" for emails).
2. **Set up for Actionable Messages in Outlook:**
   * **Provider Registration:** This is crucial for sending actionable emails. You'll need to register a "Provider" with Microsoft. This typically involves going to the [Actionable Messages Developer Dashboard](https://outlook.office.com/connectors/oam/publish).
     + Provide a Friendly Name, Sender Email Address, and Target URLs (initially, this can be a placeholder or a simple endpoint you'll build later).
     + Choose the "Test Users" scope for your POC to avoid organizational approval delays. Add your email address as a test user.
     + **Crucially, obtain the "Originator ID"** generated during this registration. This ID is a GUID that needs to be included in your actionable message payload for Outlook to render it correctly.
   * **Sender Verification (for production):** While you might skip this for a quick POC with test users, for production, understand that actionable emails require sender verification (DKIM, SPF, or Signed Cards). Signed cards are generally recommended for broader compatibility and security.
3. **Send a Basic Actionable Email:**
   * **JSON Payload:** Create a simple Adaptive Card JSON with an Action.Submit button. For your first test, the Action.Submit can just have a placeholder URL for now, or even just be a visual button without immediate backend integration.
   * **HTML Wrapper:** For Outlook, the Adaptive Card JSON needs to be embedded within a <script type="application/adaptivecard+json"></script> tag inside the HTML body of the email.
   * **Originator Header:** Include the Originator header in your email, with the GUID obtained from the provider registration.
   * **Sending Mechanism:**
     + **Power Automate (Recommended for POC):** This is the easiest way to send actionable emails without extensive coding. Use the "Send an email (V2)" action and embed the Adaptive Card JSON in the HTML body.
     + **Custom Application/API:** If you prefer coding, you can use any language/framework to send an email via SMTP and construct the HTML body with the Adaptive Card.

**Phase 2: HTTP Calls with Input Fields**

**Outcome:** Demonstrate the ability to capture user input from an Adaptive Card and send it as a payload to an external API.

**Steps:**

1. **Design the Interactive Card:**
   * In the Adaptive Card Designer, create a card with one or more Input.Text elements (e.g., for comments, a value, etc.).
   * Add an Action.Http action (or Action.Submit if using Power Automate to handle the submission).
   * **Crucially, assign id properties to your input fields.** These IDs will be used to reference the input values in the body of your HTTP POST request.
   * For Action.Http, define the url, method (POST), and body. The body will be a JSON string where you can use template syntax (e.g., {{inputId.value}}) to include the input field values.
2. **Create a Backend Endpoint (API):**
   * **Technology Choice:** You can use a simple web API framework (e.g., Node.js with Express, Python with Flask/Django, C# with ASP.NET Core, Azure Functions, AWS Lambda, Google Cloud Functions, or even a Power Automate HTTP Request trigger).
   * **Endpoint Functionality:**
     + This endpoint should be configured to receive HTTP POST requests.
     + It should parse the JSON payload sent from the Adaptive Card.
     + For the POC, simply log the received payload to verify that the input fields are being transmitted correctly. You can later integrate this with your organization's systems (e.g., update a database, trigger a workflow).
3. **Integrate and Test:**
   * Update your email sending mechanism (Power Automate or custom code) to send the new Adaptive Card JSON.
   * Ensure the url in your Action.Http points to your newly created backend endpoint.
   * Send the email, interact with the card, and verify that your backend endpoint receives the expected payload with the user's input.

**Phase 3: Security Considerations**

**Outcome:** Identify and implement basic security measures relevant to Adaptive Cards and actionable messages.

**Steps:**

1. **Sender Verification (Revisit from Phase 1):**
   * **Explain:** Emphasize the importance of DKIM/SPF or Signed Cards for production environments to prevent spoofing and ensure the authenticity of actionable messages. For the POC, simply acknowledging this is good.
   * **For production deployment, work with your Exchange/IT team** to configure SPF and DKIM for your sending domain, or explore implementing signed cards. Microsoft strongly recommends signed cards for scenarios where SPF/DKIM might fail or if you send from multiple email accounts.
2. **Webhook/API Endpoint Security:**
   * **HTTPS:** Your Action.Http URLs **must** use HTTPS. This encrypts the data in transit, protecting sensitive information.
   * **Input Validation:** On your backend API, always validate and sanitize all input received from the Adaptive Card. Never trust user input directly. This prevents injection attacks (e.g., SQL injection, cross-site scripting).
   * **Authentication/Authorization:**
     + **Microsoft Bearer Token Verification:** Requests from Microsoft (when an action is taken on an actionable message) include a bearer token in the Authorization header. Your backend endpoint should verify this token to ensure the request originated from Microsoft and, optionally, to verify the user's identity. Microsoft provides libraries and documentation for this.
     + **Custom Tokens/Correlation:** For your specific application logic, consider including a unique, limited-purpose token (e.g., a GUID or a JWT) in the url or body of your Action.Http action. This token can be generated by your service when the card is created and used by your backend to correlate the action with the original context and verify the user's authorization to perform that specific action. This adds an extra layer of security and helps prevent unauthorized actions if someone were to tamper with the URL.
     + **Least Privilege:** Ensure your backend API has only the necessary permissions to perform the requested action.
3. **Data Sensitivity:**
   * **Avoid Sensitive Data in Card JSON:** While Adaptive Cards are rendered client-side, it's generally best practice to avoid embedding highly sensitive information directly in the card's JSON, especially if the card is cached or could be forwarded.
   * **Refresh on Open (autoInvokeAction):** For scenarios where data might become stale or sensitive information needs to be up-to-date, consider using autoInvokeAction. This allows your service to retrieve an updated Adaptive Card payload when the user opens the email, ensuring they see the latest information. This also means you can avoid putting initial sensitive data in the email.
4. **Error Handling and Logging:**
   * Implement robust error handling on your backend API.
   * Log all successful and failed interactions with Adaptive Cards for auditing and troubleshooting.

**POC Deliverables:**

1. **Demonstration of a working actionable email:**
   * A simple Adaptive Card email with a button.
   * When the button is clicked, an action is triggered (e.g., a toast message in Outlook, or a successful HTTP call).
2. **Demonstration of input field submission:**
   * An Adaptive Card email with one or more input fields (e.g., text box).
   * A submit button that, when clicked, sends the input field values as a JSON payload to your backend API.
   * Logs from your backend API showing the received payload.
3. **Documentation of Security Measures:**
   * A brief write-up outlining the security considerations implemented or planned, including:
     + Confirmation of HTTPS for the action URL.
     + Mention of input validation on the backend.
     + Discussion of bearer token verification or custom token usage for authorization.
     + Acknowledgement of sender verification (DKIM/SPF/Signed Cards) for production.
4. **Adaptive Card JSON samples:** Provide the JSON for the cards used in your POC.
5. **Code snippets for backend API (if applicable):** Simple examples of how you're receiving and processing the payload.
6. **Screenshots/short video:** To visually demonstrate the functionality.

**Key Tools and Resources:**

* **Adaptive Card Designer:** <https://adaptivecards.io/designer/>
* **Adaptive Cards Documentation:** <https://adaptivecards.io/>
* **Outlook Actionable Messages Documentation:** <https://learn.microsoft.com/en-us/outlook/actionable-messages/>
* **Actionable Messages Developer Dashboard:** <https://outlook.office.com/connectors/oam/publish>
* **Power Automate:** For easy email sending and backend trigger flows.
* **Ngrok (for local development):** If your backend API is running locally, Ngrok can expose it to the internet with an HTTPS URL for testing purposes.

By following this structured approach, you'll be able to effectively showcase the capabilities of Microsoft Adaptive Cards and address the critical aspects of security in your POC. Good luck!