Flow Equivalents for Apex

This section provides high-level draft guidance to help you understand how Flow-based alternatives could be created for the Apex classes used in the workshop. These instructions offer a starting point for exploring low-code Flow alternatives to the Apex actions used in the workshop.

Caution: These flow creation instructions are not meant to be utilized during the instructor-led workshop as they have not been tested.

GetContactIdByEmail Flow Alternative to Apex Class

Flow Name

Get_Contact_Id_By_Email_Flow

Flow Type

Auto-launched Flow
(Required for use as a custom action in Agentforce)

Input Variable

Variable Name	Type	Available for Input	Description
email	Text	Yes	Email address of the contact

Flow Steps

1. Get Records: Contact by Email
    - Object: Contact
    - Filter:
        - Email = {!email}
    - Sort: (optional)
    - Limit: 1 record
    - Store output in: foundContact
2. Decision: Contact Found?
    - If foundContact is null
        - Assign output:
            - contactId = null
            - message = "No contact found with that email."
    - Else

      - Assign output:
        - contactId = {!foundContact.Id}
        - message = ""

Output Variables

Variable Name	Type	Available for Output	Description
contactId	Text	Yes	ID of the contact (or null if not found)
message	Text	Yes	Status message for the Copilot response

Note: contactId is of type Text in the flow (not ID), but behaves equivalently in Agentforce.

Agentforce Action Configuration

- Map input: email
- Map outputs:
  - contactId → downstream action (if chaining)
  - message → Copilot response, optional

GetCasesByEmail Flow Alternative to Apex Class

Flow Name

Get_Cases_By_Email_Flow

Flow Type

Auto-launched Flow
(Required for Agentforce custom actions)

Input Variables

Variable Name	Type	Available for Input	Description
email	Text	Yes	Email address of the contact

Flow Steps

1. Get Contact by Email
   - Element: Get Records
   - Object: Contact
   - Condition: Email = {!email}

- Limit: 1
    - Output: foundContact
2. Decision: Contact Found?
    - If foundContact is null → assign output message: "No contact found with that email."
    - Else → proceed to fetch cases
3. Get Related Cases
    - Element: Get Records
    - Object: Case
    - Condition: ContactId = {!foundContact.Id}
    - Fields: Id, CaseNumber, Subject, Status, Priority, CreatedDate
    - Sort by: CreatedDate DESC
    - Store all records in collection: caseList
4. Decision: Any Cases Found?
    - If caseList is empty → assign output message: "No cases found for this contact."
    - Else → continue
5. Loop: Through Cases
    - Loop over caseList
    - In each iteration, use Assignment or Text Template to format case details
    - Append each formatted entry to a Text Collection Variable: caseDescriptions
6. After Loop: Join Case Details
    - Use Text Template to join all entries into a single string
    - Assign result to caseData

Output Variable

Variable Name	Type	Available for Output	Description
caseData	Text	Yes	Final formatted case list output

Agentforce Configuration Notes

- Mark email as required input in the Agent Action setup
- Use caseData as the response variable
- Optional: add basic validation or fallback message if needed in flow

PersonalizedOpportunities Flow Alternative to Apex Class

TBD

GetCurrentUserAction Flow Alternative to Apex Class

Flow Title

Get Current User Flow

Flow Type

Auto-launched Flow
(Required for use as an Agentforce custom action)

Input Variables

None required
(Agentforce provides the current user context automatically)

Steps

1. Assignment: Store Current User Id
   - Create a new text variable: currentUserId (Available for input: unchecked)
   - Assign value: currentUserId = {!$User.Id}
2. Get Records: Get Current User
   - Object: User
   - Filter: Id = {!currentUserId}
   - Store first record only
   - Output: currentUser
3. Create Output Variable
   - Variable Name: userRecord
   - Data Type: Record
   - Object: User
   - Available for output: checked
   - Assign: userRecord = {!currentUser}

Output Variables

- userRecord (Record → User)
  This is returned to the agent for further use in downstream actions or prompts.

Notes for Agentforce Usage

- Ensure no inputs are required in the action configuration.
- Mark userRecord as the output field in the custom action setup.

- You can now use this flow to fetch current user details natively within Agentforce, replacing the Apex logic entirely.

AddCommentToCase Flow Alternative to Apex Class

Flow Type: Auto-launched Flow

→ Flow to be used in Agentforce custom actions

Step-by-Step (Agentforce-Friendly)

1. Create 3 Input Variables (API name must match the Action definition)
   - caseId (Text, Available for input)
   - contactId (Text, Available for input)
   - commentBody (Text, Available for input)
2. Get Records: Case
   - Object: Case
   - Condition:
     - Id = {!caseId}
     - ContactId = {!contactId}
   - Store first record only
   - Label: "Get Case by ID and Contact"
3. Decision: Was Case Found?
   - If Get Records result is null → Assign error message or exit flow
   - If case found → Proceed to comment creation
4. Create Records: CaseComment
   - Set fields:
     - ParentId = {!caseId}
     - CommentBody = {!commentBody}
     - IsPublished = true
   - Label: "Add Case Comment"
5. Create Output Variable
   - message (Text, Available for output)
   - Assign: "Comment added successfully to Case ID: {!caseId}"

Additional Agentforce Tips

- Name your Flow meaningfully, e.g., Add_Comment_To_Case_Flow
- In Agent Action setup:
  - Map input fields to caseId, contactId, commentBody
  - Map output to message
- Don't forget to mark variables as “Available for input/output” in the Flow