Integration Flow

# Flow Breakdown

## Step ### Integration Flow Documentation for AIS Implementation #### \*\*Integration Flow Title\*\*: Salesforce Group Management and Chatter Integration --- ### \*\*Business Overview\*\* The integration flow facilitates two primary operations: 1. \*\*GroupGetByName\*\* - Retrieves a Salesforce group ID by querying its name. 2. \*\*Chatter-FeedItem-Post and -Put\*\* - Posts or updates feed items in Salesforce Chatter. The flow is designed to integrate seamlessly with Salesforce's APIs, ensuring authentication through access tokens, error handling, and real-time logging for observability. --- ## \*\*Data Flow & System Interactions\*\* ### 1. \*\*GroupGetByName\*\* - #### \*\*Source\*\*: External HTTP Request via Azure Function HTTP trigger. - Query Parameter: Group Name (`groupName`). - #### \*\*Transformation/Steps\*\*: - Error variables are initialized at the start of the flow. - An HTTP Authorization request is sent to validate the user's session via a custom Authorization API. - Salesforce's `CollaborationGroup` object is queried to find the `ID` corresponding to the provided group name. - If the `ID` is not found, a 404 error is returned; otherwise, the `ID` is included in the response payload. - #### \*\*Outbound API/System\*\*: Salesforce SOQL Query API. - API Query: `SELECT ID FROM CollaborationGroup WHERE Name = ':groupName'`. - #### \*\*Response\*\*: - Success: JSON payload `{ "id": <group ID> }`. - Failure: HTTP 404 if no group is found. --- ### 2. \*\*Chatter-FeedItem-Post\*\* - #### \*\*Source\*\*: External HTTP Request via Azure Function HTTP trigger. - Payload contains: - `messagePieces`: Array of message segments for the Chatter feed item. - `recordIDToAddFeedItemTo`: ID of the Salesforce record to which the feed item will be added. - #### \*\*Transformation/Steps\*\*: 1. Initialize error variables. 2. Generate an OAuth access token using Salesforce credentials: - API Endpoint: `/services/oauth2/token`. - Method: POST. 3. Prepare Chatter message using the payload: - Constructs `messageSegments` and specifies `feedElementType: FeedItem`. 4. Send the Chatter message via Salesforce Chatter API: - API Endpoint: `/services/data/v47.0/chatter/feed-elements`. - #### \*\*Outbound API/System\*\*: Salesforce Chatter API. - HTTP POST to add a feed item: - JSON payload format: ```json { "body": { "messageSegments": [{...}] }, "feedElementType": "FeedItem", "subjectId": <record ID> } ``` - #### \*\*Response\*\*: - Success: HTTP 201 Created when new feed item is added successfully. - Failure: Logs and error handler to track API failures. --- ### 3. \*\*Chatter-FeedItem-Put\*\* - #### \*\*Source\*\*: External HTTP Request via Azure Function HTTP trigger. - Payload contains: - `recordIDToAddFeedItemTo`: ID of the Salesforce record. - `messagePieces`: Array of message segments for comparison with existing posts. - #### \*\*Transformation/Steps\*\*: 1. Initialize error variables. 2. Retrieve existing Chatter posts for the given `ParentID` (`recordIDToAddFeedItemTo`): - API Endpoint: `/services/data/v47.0/chatter/feed-elements`. - Query: `SELECT body FROM FeedItem WHERE ParentID = ':itemID'`. 3. Condition handling: - If the incoming message does not exist in the existing posts: - Prepare new Chatter message using the payload. - Call Salesforce Chatter API to add the post. - If the message already exists: - Skip posting and log that a duplicate post exists. - #### \*\*Outbound API/System\*\*: Salesforce Chatter API. - HTTP POST for Adding a Feed Item (same endpoint as POST flow). - HTTP GET for Existing Feed Items. - #### \*\*Response\*\*: - Success: - HTTP 201 Created if a new post is added. - HTTP 200 OK if duplicate post detected. - Failure: - Logs and error handler for tracking. --- ### 4. \*\*Salesforce Access Token Generation (Sub-flow)\*\* - #### \*\*Source\*\*: Called internally as a dependency by `Chatter-FeedItem-Post` or `Chatter-FeedItem-Put`. - #### \*\*Transformation/Steps\*\*: 1. Prepare `multipart/form-data` payload to send credentials to Salesforce OAuth API: - `client\_id`, `client\_secret`, `username`, and `password` are included. 2. Call Salesforce's `/services/oauth2/token` endpoint with the POST request. 3. Parse and set the `access\_token` from the response in the execution context. - #### \*\*Outbound API/System\*\*: Salesforce OAuth API. - HTTP POST to `/services/oauth2/token`. - #### \*\*Response\*\*: - Success: Access token stored for use in subsequent flows. - Failure: Logs and error handler for API failures during token generation. --- ## \*\*Error Handling\*\* ### Common Error Scenarios: 1. \*\*Salesforce Credential Failure\*\*: - Cause: Incorrect clientId, clientSecret, or user credentials. - Resolution: Log error with exception details; notify admin group via Application Insights alert. 2. \*\*No Group ID Found\*\*: - Cause: Non-existent `groupName` in Salesforce. - Resolution: Return 404 Not Found and log the error. 3. \*\*FeedItem Duplicate Detection\*\*: - Cause: A Chatter post matching the incoming payload already exists. - Resolution: Log the event; return HTTP 200 OK. --- ## \*\*System Design\*\* | \*\*System Component\*\* | \*\*Purpose\*\* | |---------------------------------|-------------------------------------------------------------------------------------------| | \*\*Azure HTTP Functions\*\* | Handles external triggers; defines entry point for integration flows. | | \*\*MediatR Handlers\*\* | Implements CQRS principles; decouples command/query logic from Azure Functions. | | \*\*Infrastructure Adapters\*\* | Handles communication with Salesforce APIs and external dependencies. | | \*\*Application Insights\*\* | Provides observability, error tracking, and monitoring capabilities. | | \*\*Service Bus Topics\*\* | Used for retry mechanisms, error recovery, and message passing between components. | --- ## \*\*Integration Flow Diagram\*\* #### \*\*Diagram Overview\*\*: 1️⃣ HTTP Request ➡️ 2️⃣ [Azure Function Trigger] ➡️ 3️⃣ [Mediator Command/Query] ➡️ 4️⃣ [Domain Logic] ➡️ 5️⃣ Salesforce API Integration ⬆️ Logs & Observability (Application Insights) --- This integration flow document outlines the AIS implementation with modular flows and event-driven design principles. Let me know if further clarification is needed!

Salesforce Group Management and Chatter Integration  
  
---  
  
### \*\*Business Overview\*\*  
  
The integration flow facilitates two primary operations:  
  
1. \*\*GroupGetByName\*\* - Retrieves a Salesforce group ID by querying its name.  
2. \*\*Chatter-FeedItem-Post and -Put\*\* - Posts or updates feed items in Salesforce Chatter.  
  
The flow is designed to integrate seamlessly with Salesforce's APIs, ensuring authentication through access tokens, error handling, and real-time logging for observability.  
  
---  
  
## \*\*Data Flow & System Interactions\*\*  
  
### 1. \*\*GroupGetByName\*\*  
  
- #### \*\*Source\*\*: External HTTP Request via Azure Function HTTP trigger.  
 - Query Parameter: Group Name (`groupName`).  
   
- #### \*\*Transformation/Steps\*\*:  
 - Error variables are initialized at the start of the flow.  
 - An HTTP Authorization request is sent to validate the user's session via a custom Authorization API.  
 - Salesforce's `CollaborationGroup` object is queried to find the `ID` corresponding to the provided group name.  
 - If the `ID` is not found, a 404 error is returned; otherwise, the `ID` is included in the response payload.  
  
- #### \*\*Outbound API/System\*\*: Salesforce SOQL Query API.  
 - API Query: `SELECT ID FROM CollaborationGroup WHERE Name = ':groupName'`.  
  
- #### \*\*Response\*\*:  
 - Success: JSON payload `{ "id": <group ID> }`.  
 - Failure: HTTP 404 if no group is found.  
  
---  
  
### 2. \*\*Chatter-FeedItem-Post\*\*  
  
- #### \*\*Source\*\*: External HTTP Request via Azure Function HTTP trigger.  
 - Payload contains:  
 - `messagePieces`: Array of message segments for the Chatter feed item.  
 - `recordIDToAddFeedItemTo`: ID of the Salesforce record to which the feed item will be added.  
  
- #### \*\*Transformation/Steps\*\*:  
 1. Initialize error variables.  
 2. Generate an OAuth access token using Salesforce credentials:  
 - API Endpoint: `/services/oauth2/token`.  
 - Method: POST.  
 3. Prepare Chatter message using the payload:  
 - Constructs `messageSegments` and specifies `feedElementType: FeedItem`.  
 4. Send the Chatter message via Salesforce Chatter API:  
 - API Endpoint: `/services/data/v47.0/chatter/feed-elements`.  
  
- #### \*\*Outbound API/System\*\*: Salesforce Chatter API.  
 - HTTP POST to add a feed item:  
 - JSON payload format:  
 ```json  
 {  
 "body": {  
 "messageSegments": [{...}]  
 },  
 "feedElementType": "FeedItem",  
 "subjectId": <record ID>  
 }  
 ```  
  
- #### \*\*Response\*\*:  
 - Success: HTTP 201 Created when new feed item is added successfully.  
 - Failure: Logs and error handler to track API failures.  
  
---  
  
### 3. \*\*Chatter-FeedItem-Put\*\*  
  
- #### \*\*Source\*\*: External HTTP Request via Azure Function HTTP trigger.  
 - Payload contains:  
 - `recordIDToAddFeedItemTo`: ID of the Salesforce record.  
 - `messagePieces`: Array of message segments for comparison with existing posts.  
  
- #### \*\*Transformation/Steps\*\*:  
 1. Initialize error variables.  
 2. Retrieve existing Chatter posts for the given `ParentID` (`recordIDToAddFeedItemTo`):  
 - API Endpoint: `/services/data/v47.0/chatter/feed-elements`.  
 - Query: `SELECT body FROM FeedItem WHERE ParentID = ':itemID'`.  
 3. Condition handling:  
 - If the incoming message does not exist in the existing posts:  
 - Prepare new Chatter message using the payload.  
 - Call Salesforce Chatter API to add the post.  
 - If the message already exists:  
 - Skip posting and log that a duplicate post exists.  
  
- #### \*\*Outbound API/System\*\*: Salesforce Chatter API.  
 - HTTP POST for Adding a Feed Item (same endpoint as POST flow).  
 - HTTP GET for Existing Feed Items.  
  
- #### \*\*Response\*\*:  
 - Success:  
 - HTTP 201 Created if a new post is added.  
 - HTTP 200 OK if duplicate post detected.  
 - Failure:  
 - Logs and error handler for tracking.  
  
---  
  
### 4. \*\*Salesforce Access Token Generation (Sub-flow)\*\*  
  
- #### \*\*Source\*\*: Called internally as a dependency by `Chatter-FeedItem-Post` or `Chatter-FeedItem-Put`.  
  
- #### \*\*Transformation/Steps\*\*:  
 1. Prepare `multipart/form-data` payload to send credentials to Salesforce OAuth API:  
 - `client\_id`, `client\_secret`, `username`, and `password` are included.  
 2. Call Salesforce's `/services/oauth2/token` endpoint with the POST request.  
 3. Parse and set the `access\_token` from the response in the execution context.  
  
- #### \*\*Outbound API/System\*\*: Salesforce OAuth API.  
 - HTTP POST to `/services/oauth2/token`.  
  
- #### \*\*Response\*\*:  
 - Success: Access token stored for use in subsequent flows.  
 - Failure: Logs and error handler for API failures during token generation.  
  
---  
  
## \*\*Error Handling\*\*  
  
### Common Error Scenarios:  
1. \*\*Salesforce Credential Failure\*\*:  
 - Cause: Incorrect clientId, clientSecret, or user credentials.  
 - Resolution: Log error with exception details; notify admin group via Application Insights alert.  
  
2. \*\*No Group ID Found\*\*:  
 - Cause: Non-existent `groupName` in Salesforce.  
 - Resolution: Return 404 Not Found and log the error.  
  
3. \*\*FeedItem Duplicate Detection\*\*:  
 - Cause: A Chatter post matching the incoming payload already exists.  
 - Resolution: Log the event; return HTTP 200 OK.  
  
---  
  
## \*\*System Design\*\*  
  
| \*\*System Component\*\* | \*\*Purpose\*\* |  
|---------------------------------|-------------------------------------------------------------------------------------------|  
| \*\*Azure HTTP Functions\*\* | Handles external triggers; defines entry point for integration flows. |  
| \*\*MediatR Handlers\*\* | Implements CQRS principles; decouples command/query logic from Azure Functions. |  
| \*\*Infrastructure Adapters\*\* | Handles communication with Salesforce APIs and external dependencies. |  
| \*\*Application Insights\*\* | Provides observability, error tracking, and monitoring capabilities. |  
| \*\*Service Bus Topics\*\* | Used for retry mechanisms, error recovery, and message passing between components. |  
  
---  
  
## \*\*Integration Flow Diagram\*\*  
#### \*\*Diagram Overview\*\*:  
1️⃣ HTTP Request ➡️ 2️⃣ [Azure Function Trigger] ➡️ 3️⃣ [Mediator Command/Query] ➡️ 4️⃣ [Domain Logic] ➡️ 5️⃣ Salesforce API Integration   
⬆️ Logs & Observability (Application Insights)  
  
---  
  
This integration flow document outlines the AIS implementation with modular flows and event-driven design principles. Let me know if further clarification is needed!