Integration Flow

# Flow Breakdown

## Step Below, I have developed Integration Flow Documentation for the AIS (Application Integration Solutions) project that aligns with the translated architecture from the MuleSoft implementation. --- ## Integration Flow Documentation for AIS Salesforce Integration ### \*\*1. Overview\*\* This document explains the integration flows designed for AIS Salesforce Integration. These flows handle: 1. Retrieving collaboration group information by name. 2. Posting FeedItem (Chatter message) to Salesforce. 3. Updating an existing FeedItem in Salesforce. The flows are designed to use Azure Functions, Salesforce APIs, and Azure Service Bus to provide scalable, secure, and event-driven integration. Each flow demonstrates specific data transformations, interactions, and responsibilities. --- ### \*\*2. Integration Flows\*\* | Flow Name | Description | |------------------------- |---------------------------------------------------------------------------------------------------------| | \*\*Group-GetByName\*\* | Fetches the ID of the requested Collaboration Group in Salesforce using the group's name. | | \*\*Chatter-FeedItem-Post\*\* | Posts a new Chatter message (FeedItem) in Salesforce for a specific Salesforce record. | | \*\*Chatter-FeedItem-Put\*\* | Updates or posts Chatter messages only if a check determines no such message exists for the record. | --- ### \*\*3. Detailed Flow Descriptions\*\* #### \*\*Flow 1: Group-GetByName\*\* - \*\*Trigger\*\* - \*\*HTTP Request\*\* or \*\*Message Received\*\* from `Group.GetByName.Request` Service Bus Topic. - Input: ```json { "groupName": "Sample Group Name" } ``` - \*\*Processing\*\* 1. Validate that the `groupName` parameter exists in the request. 2. Log the operation for monitoring purposes. 3. Query Salesforce's `CollaborationGroup` object with the name provided via Salesforce’s `Query` API: Query: `SELECT ID FROM CollaborationGroup WHERE Name = ':groupName'`. 4. Process the result payload: - If the group is found, return the group ID. - If the group is not found, return a `404` HTTP status code. - \*\*Outbound Call\*\* - \*\*API\*\*: Salesforce SOQL Query API. - \*\*Request Payload\*\*: Salesforce API binds `groupName` using query parameters. - \*\*Response\*\*: ```json { "id": "001XXXXXXXXXXXXXXX" } ``` --- #### \*\*Flow 2: Chatter-FeedItem-Post\*\* - \*\*Trigger\*\* - \*\*HTTP Request\*\* or \*\*Message Received\*\* from `Chatter.FeedItem.Post.Request` Service Bus Topic. - Input: ```json { "recordIDToAddFeedItemTo": "001XXXXXXXXXXXXXXX", "messagePieces": [ { "type": "Text", "text": "This is a sample Chatter message." } ] } ``` - \*\*Processing\*\* 1. Generate an OAuth Access Token by calling the \*\*GenerateChatterAccessToken\*\* sub-flow. 2. Create a structured Chatter message payload using the provided data, structured as: ```json { "body": { "messageSegments": [ { "type": "Text", "text": "This is a sample Chatter message." } ] }, "feedElementType": "FeedItem", "subjectId": "001XXXXXXXXXXXXXXX" } ``` 3. Call Salesforce Chatter Feed API to post the FeedItem: - Endpoint: `/services/data/v47.0/chatter/feed-elements`. - \*\*Outbound Call\*\* - \*\*API\*\*: Salesforce Chatter Feed API. - \*\*Request Payload\*\*: Prepared Chatter message as shown above. - \*\*Response\*\*: No content (201 Created). --- #### \*\*Flow 3: Chatter-FeedItem-Put\*\* - \*\*Trigger\*\* - \*\*HTTP Request\*\* or \*\*Message Received\*\* from `Chatter.FeedItem.Put.Request` Service Bus Topic. - Input: ```json { "recordIDToAddFeedItemTo": "001XXXXXXXXXXXXXXX", "messagePieces": [ { "type": "Text", "text": "This is the updated Chatter message." } ] } ``` - \*\*Processing\*\* 1. Generate an OAuth Access Token by calling the \*\*GenerateChatterAccessToken\*\* sub-flow. 2. Check for existing Chatter messages (FeedItem) using: - Query on `FeedItem` object: `SELECT body FROM FeedItem WHERE ParentID = ':recordIDToAddFeedItemTo'`. 3. If an identical message already exists: - Log the message and skip creation. 4. If no message exists: - Prepare Chatter message payload (as in Flow 2) and send a POST request. - \*\*Outbound Call\*\* - \*\*API\*\*: Salesforce Chatter Feed API. - \*\*Request Payload\*\*: Prepared Chatter message. - \*\*Response\*\*: - 200 OK (Skipped) or - 201 Created (New Post). --- #### \*\*Sub-Flow: GenerateChatterAccessToken\*\* - \*\*Description\*\* Handles the generation of an OAuth Access Token required for Salesforce API calls. - \*\*Integrated APIs\*\* - \*\*Endpoint\*\*: `/services/oauth2/token` - \*\*Request Payload\*\*: Form-data with credentials: ```json { "grant\_type": "password", "client\_id": "xxxxxxxxxxxx", "client\_secret": "xxxxxxxxxxxx", "username": "sample\_username", "password": "sample\_password" } ``` - \*\*Response\*\*: ```json { "access\_token": "Bearer xxxxxxxxxxx" } ``` - \*\*Reusability\*\* Called by all flows requiring Salesforce API access. --- ### \*\*4. Service Bus Topology\*\* | Topic Name | Message Type | Example Payload Key | |--------------------------------|--------------|------------------------------| | `Group.GetByName.Request` | Command | `"groupName"` | | `Chatter.FeedItem.Post.Request` | Command | `"recordIDToAddFeedItemTo"` | | `Chatter.FeedItem.Put.Request` | Command | `"recordIDToAddFeedItemTo"` | --- ### \*\*5. Error Handling & Logging\*\* #### \*\*Error Notifications\*\* - An error handling strategy is applied globally: - Sets error details in a standard format. - Sends notifications if critical errors occur. #### \*\*Logging\*\* - Logs are captured at each critical step: - Before/after HTTP requests. - Before processing messages from Service Bus. - Before and after Salesforce API calls. --- ### \*\*6. Security\*\* - \*\*OAuth Authentication\*\*: - All Salesforce API calls require an access token fetched from the `GenerateChatterAccessToken` sub-flow. - \*\*Configuration Management\*\*: - Credentials and sensitive data are stored securely in the `local.settings.json` file. --- This comprehensive integration flow documentation ensures stakeholders understand the interaction points, data transformations, and service integration for each use case. Let me know if you'd like further assistance!

1. Retrieving collaboration group information by name.  
2. Posting FeedItem (Chatter message) to Salesforce.  
3. Updating an existing FeedItem in Salesforce.  
  
The flows are designed to use Azure Functions, Salesforce APIs, and Azure Service Bus to provide scalable, secure, and event-driven integration. Each flow demonstrates specific data transformations, interactions, and responsibilities.  
  
---  
  
### \*\*2. Integration Flows\*\*  
  
| Flow Name | Description |  
|------------------------- |---------------------------------------------------------------------------------------------------------|  
| \*\*Group-GetByName\*\* | Fetches the ID of the requested Collaboration Group in Salesforce using the group's name. |  
| \*\*Chatter-FeedItem-Post\*\* | Posts a new Chatter message (FeedItem) in Salesforce for a specific Salesforce record. |  
| \*\*Chatter-FeedItem-Put\*\* | Updates or posts Chatter messages only if a check determines no such message exists for the record. |  
  
---  
  
### \*\*3. Detailed Flow Descriptions\*\*  
  
#### \*\*Flow 1: Group-GetByName\*\*  
  
- \*\*Trigger\*\*  
 - \*\*HTTP Request\*\* or \*\*Message Received\*\* from `Group.GetByName.Request` Service Bus Topic.  
 - Input:   
 ```json  
 {  
 "groupName": "Sample Group Name"  
 }  
 ```  
  
- \*\*Processing\*\*  
 1. Validate that the `groupName` parameter exists in the request.  
 2. Log the operation for monitoring purposes.  
 3. Query Salesforce's `CollaborationGroup` object with the name provided via Salesforce’s `Query` API:   
 Query: `SELECT ID FROM CollaborationGroup WHERE Name = ':groupName'`.  
 4. Process the result payload:  
 - If the group is found, return the group ID.  
 - If the group is not found, return a `404` HTTP status code.  
  
- \*\*Outbound Call\*\*  
 - \*\*API\*\*: Salesforce SOQL Query API.  
 - \*\*Request Payload\*\*: Salesforce API binds `groupName` using query parameters.  
 - \*\*Response\*\*:   
 ```json  
 {  
 "id": "001XXXXXXXXXXXXXXX"  
 }  
 ```  
  
---  
  
#### \*\*Flow 2: Chatter-FeedItem-Post\*\*  
  
- \*\*Trigger\*\*  
 - \*\*HTTP Request\*\* or \*\*Message Received\*\* from `Chatter.FeedItem.Post.Request` Service Bus Topic.  
 - Input:   
 ```json  
 {  
 "recordIDToAddFeedItemTo": "001XXXXXXXXXXXXXXX",  
 "messagePieces": [  
 {  
 "type": "Text",  
 "text": "This is a sample Chatter message."  
 }  
 ]  
 }  
 ```  
  
- \*\*Processing\*\*  
 1. Generate an OAuth Access Token by calling the \*\*GenerateChatterAccessToken\*\* sub-flow.  
 2. Create a structured Chatter message payload using the provided data, structured as:   
 ```json  
 {  
 "body": {  
 "messageSegments": [  
 {  
 "type": "Text",  
 "text": "This is a sample Chatter message."  
 }  
 ]  
 },  
 "feedElementType": "FeedItem",  
 "subjectId": "001XXXXXXXXXXXXXXX"  
 }  
 ```  
 3. Call Salesforce Chatter Feed API to post the FeedItem:  
 - Endpoint: `/services/data/v47.0/chatter/feed-elements`.  
  
- \*\*Outbound Call\*\*  
 - \*\*API\*\*: Salesforce Chatter Feed API.  
 - \*\*Request Payload\*\*: Prepared Chatter message as shown above.  
 - \*\*Response\*\*: No content (201 Created).  
  
---  
  
#### \*\*Flow 3: Chatter-FeedItem-Put\*\*  
  
- \*\*Trigger\*\*  
 - \*\*HTTP Request\*\* or \*\*Message Received\*\* from `Chatter.FeedItem.Put.Request` Service Bus Topic.  
 - Input:  
 ```json  
 {  
 "recordIDToAddFeedItemTo": "001XXXXXXXXXXXXXXX",  
 "messagePieces": [  
 {  
 "type": "Text",  
 "text": "This is the updated Chatter message."  
 }  
 ]  
 }  
 ```  
  
- \*\*Processing\*\*  
 1. Generate an OAuth Access Token by calling the \*\*GenerateChatterAccessToken\*\* sub-flow.  
 2. Check for existing Chatter messages (FeedItem) using:  
 - Query on `FeedItem` object: `SELECT body FROM FeedItem WHERE ParentID = ':recordIDToAddFeedItemTo'`.  
 3. If an identical message already exists:  
 - Log the message and skip creation.  
 4. If no message exists:  
 - Prepare Chatter message payload (as in Flow 2) and send a POST request.  
  
- \*\*Outbound Call\*\*  
 - \*\*API\*\*: Salesforce Chatter Feed API.  
 - \*\*Request Payload\*\*: Prepared Chatter message.  
 - \*\*Response\*\*:  
 - 200 OK (Skipped) or  
 - 201 Created (New Post).  
  
---  
  
#### \*\*Sub-Flow: GenerateChatterAccessToken\*\*  
  
- \*\*Description\*\*  
 Handles the generation of an OAuth Access Token required for Salesforce API calls.  
   
- \*\*Integrated APIs\*\*  
 - \*\*Endpoint\*\*: `/services/oauth2/token`  
 - \*\*Request Payload\*\*: Form-data with credentials:  
 ```json  
 {  
 "grant\_type": "password",  
 "client\_id": "xxxxxxxxxxxx",  
 "client\_secret": "xxxxxxxxxxxx",  
 "username": "sample\_username",  
 "password": "sample\_password"  
 }  
 ```  
 - \*\*Response\*\*:  
 ```json  
 {  
 "access\_token": "Bearer xxxxxxxxxxx"  
 }  
 ```  
  
- \*\*Reusability\*\*  
 Called by all flows requiring Salesforce API access.  
  
---  
  
### \*\*4. Service Bus Topology\*\*  
  
| Topic Name | Message Type | Example Payload Key |  
|--------------------------------|--------------|------------------------------|  
| `Group.GetByName.Request` | Command | `"groupName"` |  
| `Chatter.FeedItem.Post.Request` | Command | `"recordIDToAddFeedItemTo"` |  
| `Chatter.FeedItem.Put.Request` | Command | `"recordIDToAddFeedItemTo"` |  
  
---  
  
### \*\*5. Error Handling & Logging\*\*  
  
#### \*\*Error Notifications\*\*  
- An error handling strategy is applied globally:  
 - Sets error details in a standard format.  
 - Sends notifications if critical errors occur.  
  
#### \*\*Logging\*\*  
- Logs are captured at each critical step:  
 - Before/after HTTP requests.  
 - Before processing messages from Service Bus.  
 - Before and after Salesforce API calls.  
  
---  
  
### \*\*6. Security\*\*  
  
- \*\*OAuth Authentication\*\*:  
 - All Salesforce API calls require an access token fetched from the `GenerateChatterAccessToken` sub-flow.  
- \*\*Configuration Management\*\*:  
 - Credentials and sensitive data are stored securely in the `local.settings.json` file.  
  
---  
  
This comprehensive integration flow documentation ensures stakeholders understand the interaction points, data transformations, and service integration for each use case. Let me know if you'd like further assistance!