Integration Flow

# Flow Breakdown

## Step Here is the \*\*Integration Flow Documentation\*\* for the AIS project based on the architecture implementation plan: --- ### \*\*Integration Flow Documentation\*\* --- #### \*\*Flow Name: Group-GetByName\*\* \*\*Purpose\*\*: This flow retrieves the Salesforce Group ID based on the group name provided via an HTTP GET request. | \*\*Step\*\* | \*\*Details\*\* | |----------|-----------------------------------------------------------------------------| | \*\*Trigger\*\* | HTTP GET: `group/get-by-name` | | \*\*Action 1\*\* | Retrieve `groupName` from the query parameters of the HTTP request. | | \*\*Action 2\*\* | Validate query parameters ensuring `groupName` is provided. | | \*\*Action 3\*\* | MediatR sends the `GetByNameCommand` to the handler. | | \*\*Logic\*\* | The handler calls the `SalesforceAdapter` to query Salesforce for the group name. | | \*\*Transformation\*\* | If the group is found, output is returned as JSON `{ id: groupId }`. If not found, it logs the error and returns a `404 Not Found` status. | | \*\*External Interaction\*\* | Salesforce API: `GET /services/data/v47.0/sobjects/CollaborationGroup` | | \*\*Logging and Monitoring\*\* | Application Insights logs errors and request metrics for observability. Email alerts for critical issues. | --- #### \*\*Flow Name: Chatter-FeedItem-Post\*\* \*\*Purpose\*\*: This flow posts a Chatter message in Salesforce associated with a specific record. | \*\*Step\*\* | \*\*Details\*\* | |----------|-----------------------------------------------------------------------------| | \*\*Trigger\*\* | HTTP POST: `chatter/post-feed-item` | | \*\*Action 1\*\* | Extract payload from the HTTP request containing `recordId` and `messageSegments`. | | \*\*Action 2\*\* | Prepare the Salesforce Chatter message payload using the handler. | | \*\*Action 3\*\* | MediatR sends the `PostFeedItemCommand` to the handler. | | \*\*Logic\*\* | Handler prepares the Chatter payload using an `Azure Function` and posts it through the `SalesforceAdapter`. | | \*\*Transformation\*\* | The payload is structured into JSON format containing `recordId`, `messageSegments`, and `feedElementType: FeedItem`. | | \*\*External Interaction\*\* | Salesforce API: `POST /services/data/v47.0/chatter/feed-elements` | | \*\*Logging and Monitoring\*\* | Logs the payload before and after sending to the Salesforce API. Monitors failures with Application Insights. | --- #### \*\*Flow Name: Chatter-FeedItem-Put\*\* \*\*Purpose\*\*: Updates or conditionally adds a Chatter post to Salesforce, ensuring duplicates are avoided. | \*\*Step\*\* | \*\*Details\*\* | |----------|-----------------------------------------------------------------------------| | \*\*Trigger\*\* | HTTP PUT: `chatter/put-feed-item` | | \*\*Action 1\*\* | Extract payload from the HTTP request containing `recordId` and message content. | | \*\*Action 2\*\* | Use SalesforceAdapter via the handler to query existing posts for the `recordId`. | | \*\*Logic 1\*\* | If similar posts exist, skip the creation of a new message. Otherwise, prepare and post the message. | | \*\*Transformation\*\* | Conditional payload creation: skips duplicate posts ensuring consistency with Salesforce records. | | \*\*External Interaction\*\* | Salesforce API: `POST /services/data/v47.0/chatter/feed-elements` for posting messages, `GET /services/data/v47.0/feed-item` for existing post queries. | | \*\*Logging and Monitoring\*\* | Logs information for duplicates and sends monitoring data to Application Insights. | --- #### \*\*Flow Name: GenerateAccessToken\*\* \*\*Purpose\*\*: Generates Salesforce OAuth access tokens for authenticating other flows. | \*\*Step\*\* | \*\*Details\*\* | |----------|-----------------------------------------------------------------------------| | \*\*Trigger\*\* | Sub-flow call: generates an access token. | | \*\*Action 1\*\* | Constructs a multi-part form-data payload containing `grant\_type`, `client\_id`, `client\_secret`, `username`, and `password`. | | \*\*Action 2\*\* | Sends the form-data request to Salesforce OAuth Token endpoint through Azure Function. | | \*\*Logic\*\* | Handles token creation retries (reconnect) and injects generated `access\_token` into dependent functions. | | \*\*External Interaction\*\* | Salesforce API: `POST /services/oauth2/token` | | \*\*Logging and Monitoring\*\* | Logs token generation request/response metrics. Monitors failures with Application Insights. | --- ### \*\*Error Handling\*\* \*\*Global Exception Strategy:\*\* - Logs errors into Application Insights. - Sends email notifications for specific critical exceptions (e.g., authentication failures). - Ensures retries for operations prone to transient failures (e.g., Salesforce API calls). --- ### \*\*Data Flow Summary\*\* 1. \*\*Input:\*\* Payloads and query params are injected through HTTP triggers on Azure Functions. 2. \*\*Processing:\*\* Business logic is managed by MediatR commands/handlers, and adapters interact with external APIs. 3. \*\*Output:\*\* Transformed responses use consistent JSON formats for clients. 4. \*\*External System Interaction:\*\* All external interactions (Salesforce APIs) are mediated through typed adapters using interfaces to modularize endpoint communication. --- ### \*\*Design Principles\*\* - \*\*Modularity:\*\* Each operation is split into Azure Functions, adhering to CQRS and separation of concerns principles. - \*\*Scalability:\*\* Functions are stateless, enabling horizontal scaling under load. - \*\*Security:\*\* Sensitive credentials and tokens are handled using Azure Key Vault or environment variables. - \*\*Observability:\*\* Integrated Application Insights logging ensures monitoring across all flows. --- Let me know if you need diagrams or additional refinements!

---  
  
### \*\*Integration Flow Documentation\*\*  
  
---  
  
#### \*\*Flow Name: Group-GetByName\*\*  
  
\*\*Purpose\*\*: This flow retrieves the Salesforce Group ID based on the group name provided via an HTTP GET request.  
  
| \*\*Step\*\* | \*\*Details\*\* |  
|----------|-----------------------------------------------------------------------------|  
| \*\*Trigger\*\* | HTTP GET: `group/get-by-name` |  
| \*\*Action 1\*\* | Retrieve `groupName` from the query parameters of the HTTP request. |  
| \*\*Action 2\*\* | Validate query parameters ensuring `groupName` is provided. |  
| \*\*Action 3\*\* | MediatR sends the `GetByNameCommand` to the handler. |  
| \*\*Logic\*\* | The handler calls the `SalesforceAdapter` to query Salesforce for the group name. |  
| \*\*Transformation\*\* | If the group is found, output is returned as JSON `{ id: groupId }`. If not found, it logs the error and returns a `404 Not Found` status. |  
| \*\*External Interaction\*\* | Salesforce API: `GET /services/data/v47.0/sobjects/CollaborationGroup` |  
| \*\*Logging and Monitoring\*\* | Application Insights logs errors and request metrics for observability. Email alerts for critical issues. |  
  
---  
  
#### \*\*Flow Name: Chatter-FeedItem-Post\*\*  
  
\*\*Purpose\*\*: This flow posts a Chatter message in Salesforce associated with a specific record.  
  
| \*\*Step\*\* | \*\*Details\*\* |  
|----------|-----------------------------------------------------------------------------|  
| \*\*Trigger\*\* | HTTP POST: `chatter/post-feed-item` |  
| \*\*Action 1\*\* | Extract payload from the HTTP request containing `recordId` and `messageSegments`. |  
| \*\*Action 2\*\* | Prepare the Salesforce Chatter message payload using the handler. |  
| \*\*Action 3\*\* | MediatR sends the `PostFeedItemCommand` to the handler. |  
| \*\*Logic\*\* | Handler prepares the Chatter payload using an `Azure Function` and posts it through the `SalesforceAdapter`. |  
| \*\*Transformation\*\* | The payload is structured into JSON format containing `recordId`, `messageSegments`, and `feedElementType: FeedItem`. |  
| \*\*External Interaction\*\* | Salesforce API: `POST /services/data/v47.0/chatter/feed-elements` |  
| \*\*Logging and Monitoring\*\* | Logs the payload before and after sending to the Salesforce API. Monitors failures with Application Insights. |  
  
---  
  
#### \*\*Flow Name: Chatter-FeedItem-Put\*\*  
  
\*\*Purpose\*\*: Updates or conditionally adds a Chatter post to Salesforce, ensuring duplicates are avoided.  
  
| \*\*Step\*\* | \*\*Details\*\* |  
|----------|-----------------------------------------------------------------------------|  
| \*\*Trigger\*\* | HTTP PUT: `chatter/put-feed-item` |  
| \*\*Action 1\*\* | Extract payload from the HTTP request containing `recordId` and message content. |  
| \*\*Action 2\*\* | Use SalesforceAdapter via the handler to query existing posts for the `recordId`. |  
| \*\*Logic 1\*\* | If similar posts exist, skip the creation of a new message. Otherwise, prepare and post the message. |  
| \*\*Transformation\*\* | Conditional payload creation: skips duplicate posts ensuring consistency with Salesforce records. |  
| \*\*External Interaction\*\* | Salesforce API: `POST /services/data/v47.0/chatter/feed-elements` for posting messages, `GET /services/data/v47.0/feed-item` for existing post queries. |  
| \*\*Logging and Monitoring\*\* | Logs information for duplicates and sends monitoring data to Application Insights. |  
  
---  
  
#### \*\*Flow Name: GenerateAccessToken\*\*  
  
\*\*Purpose\*\*: Generates Salesforce OAuth access tokens for authenticating other flows.  
  
| \*\*Step\*\* | \*\*Details\*\* |  
|----------|-----------------------------------------------------------------------------|  
| \*\*Trigger\*\* | Sub-flow call: generates an access token. |  
| \*\*Action 1\*\* | Constructs a multi-part form-data payload containing `grant\_type`, `client\_id`, `client\_secret`, `username`, and `password`. |  
| \*\*Action 2\*\* | Sends the form-data request to Salesforce OAuth Token endpoint through Azure Function. |  
| \*\*Logic\*\* | Handles token creation retries (reconnect) and injects generated `access\_token` into dependent functions. |  
| \*\*External Interaction\*\* | Salesforce API: `POST /services/oauth2/token` |  
| \*\*Logging and Monitoring\*\* | Logs token generation request/response metrics. Monitors failures with Application Insights. |  
  
---  
  
### \*\*Error Handling\*\*  
\*\*Global Exception Strategy:\*\*   
- Logs errors into Application Insights.   
- Sends email notifications for specific critical exceptions (e.g., authentication failures).   
- Ensures retries for operations prone to transient failures (e.g., Salesforce API calls).   
  
---  
  
### \*\*Data Flow Summary\*\*  
1. \*\*Input:\*\* Payloads and query params are injected through HTTP triggers on Azure Functions.   
2. \*\*Processing:\*\* Business logic is managed by MediatR commands/handlers, and adapters interact with external APIs.   
3. \*\*Output:\*\* Transformed responses use consistent JSON formats for clients.   
4. \*\*External System Interaction:\*\* All external interactions (Salesforce APIs) are mediated through typed adapters using interfaces to modularize endpoint communication.  
  
---  
  
### \*\*Design Principles\*\*  
- \*\*Modularity:\*\* Each operation is split into Azure Functions, adhering to CQRS and separation of concerns principles.   
- \*\*Scalability:\*\* Functions are stateless, enabling horizontal scaling under load.   
- \*\*Security:\*\* Sensitive credentials and tokens are handled using Azure Key Vault or environment variables.   
- \*\*Observability:\*\* Integrated Application Insights logging ensures monitoring across all flows.   
  
---  
  
Let me know if you need diagrams or additional refinements!