Integration Flow

To generate an Azure Integration Services (AIS) project based on the outlined architecture for your integration with Salesforce, we can break down the project into different Azure Functions, workflows, and necessary configuration files. Below is the structure for an AIS project along with the code snippets for each Azure Function.  
  
### Project Structure  
  
```  
AIS.Integration.Salesforce  
│  
├── AIS.Integration.Salesforce.sln (Solution file)  
│  
├── AIS.Integration.Salesforce.Functions  
│ ├── Functions  
│ │ ├── GroupGetByName.cs  
│ │ ├── ChatterFeedItemPost.cs  
│ │ ├── ChatterFeedItemPut.cs  
│ │ ├── GenerateChatterAccessToken.cs  
│ ├── appsettings.json  
│ └── local.settings.json  
│  
├── AIS.Integration.Salesforce.di (Dependency Injection project, if necessary)  
│ └── Services  
│ ├── SalesforceService.cs  
│ └── HttpClientFactory.cs  
│  
└── AIS.Integration.Salesforce.Models  
 ├── GroupIdResponse.cs  
 └── ChatterMessage.cs  
```  
  
### Code Snippets for Azure Functions  
  
#### 1. \*\*GroupGetByName.cs\*\*  
  
```csharp  
using System.IO;  
using System.Net;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Azure.WebJobs;  
using Microsoft.Azure.WebJobs.Extensions.Http;  
using Microsoft.AspNetCore.Http;  
using Microsoft.Extensions.Logging;  
using Newtonsoft.Json;  
  
public static class GroupGetByName  
{  
 [FunctionName("GroupGetByName")]  
 public static async Task<IActionResult> Run(  
 [HttpTrigger(AuthorizationLevel.Function, "get", Route = "group/getbyname")] HttpRequest req,  
 ILogger log)  
 {  
 string groupName = req.Query["groupName"];  
   
 // Call to authorize user and retrieve group ID logic here  
   
 log.LogInformation($"Retrieving group ID for name: {groupName}");  
   
 // Assume we called another service and retrieved the ID  
 string groupId = ""; // replace with actual logic  
   
 return new OkObjectResult(new { id = groupId });  
 }  
}  
```  
  
#### 2. \*\*ChatterFeedItemPost.cs\*\*  
  
```csharp  
using System.IO;  
using System.Net;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Azure.WebJobs;  
using Microsoft.Azure.WebJobs.Extensions.Http;  
using Microsoft.AspNetCore.Http;  
using Microsoft.Extensions.Logging;  
using Newtonsoft.Json;  
  
public static class ChatterFeedItemPost  
{  
 [FunctionName("ChatterFeedItemPost")]  
 public static async Task<IActionResult> Run(  
 [HttpTrigger(AuthorizationLevel.Function, "post", Route = "chatter/feeditem/post")] HttpRequest req,  
 ILogger log)  
 {  
 string requestBody = await new StreamReader(req.Body).ReadToEndAsync();  
 var chatterMessage = JsonConvert.DeserializeObject<ChatterMessage>(requestBody);  
   
 // Prepare and post to Chatter logic here  
  
 log.LogInformation("Posting Chatter Message");  
   
 return new OkObjectResult(new { status = "Chatter message created" });  
 }  
}  
```  
  
#### 3. \*\*ChatterFeedItemPut.cs\*\*  
  
```csharp  
using System.IO;  
using System.Net;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Azure.WebJobs;  
using Microsoft.Azure.WebJobs.Extensions.Http;  
using Microsoft.AspNetCore.Http;  
using Microsoft.Extensions.Logging;  
using Newtonsoft.Json;  
  
public static class ChatterFeedItemPut  
{  
 [FunctionName("ChatterFeedItemPut")]  
 public static async Task<IActionResult> Run(  
 [HttpTrigger(AuthorizationLevel.Function, "put", Route = "chatter/feeditem/put")] HttpRequest req,  
 ILogger log)  
 {  
 string requestBody = await new StreamReader(req.Body).ReadToEndAsync();  
 var chatterMessage = JsonConvert.DeserializeObject<ChatterMessage>(requestBody);  
   
 // Logic to update Chatter Feed Item  
  
 log.LogInformation("Updating Chatter Feed Item");  
   
 return new OkObjectResult(new { status = "Chatter message updated/exists" });  
 }  
}  
```  
  
#### 4. \*\*GenerateChatterAccessToken.cs\*\*  
  
```csharp  
using System.IO;  
using System.Net;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Azure.WebJobs;  
using Microsoft.Azure.WebJobs.Extensions.Http;  
using Microsoft.AspNetCore.Http;  
using Microsoft.Extensions.Logging;  
  
public static class GenerateChatterAccessToken  
{  
 [FunctionName("GenerateChatterAccessToken")]  
 public static async Task<IActionResult> Run(  
 [HttpTrigger(AuthorizationLevel.Function, "post", Route = "chatter/accesstoken")] HttpRequest req,  
 ILogger log)  
 {  
 // Logic to generate Chatter access token  
   
 log.LogInformation("Generating Chatter Access Token");  
   
 return new OkObjectResult(new { accessToken = "your\_access\_token" });  
 }  
}  
```  
  
### Configuration Files  
  
#### \*\*appsettings.json\*\*  
  
```json  
{  
 "Salesforce": {  
 "ClientID": "your\_client\_id",  
 "ClientSecret": "your\_client\_secret",  
 "Username": "your\_username",  
 "Password": "your\_password"  
 }  
}  
```  
  
#### \*\*local.settings.json\*\*  
  
```json  
{  
 "IsEncrypted": false,  
 "Values": {  
 "AzureWebJobsStorage": "your\_azure\_webjobs\_storage\_connection\_string",  
 "FUNCTIONS\_WORKER\_RUNTIME": "dotnet",  
 "Salesforce:ClientID": "your\_client\_id",  
 "Salesforce:ClientSecret": "your\_client\_secret",  
 "Salesforce:Username": "your\_username",  
 "Salesforce:Password": "your\_password"  
 }  
}  
```  
  
### Deployment  
  
1. \*\*Build the Project:\*\* Ensure all dependencies are included, and the project builds successfully.  
2. \*\*Deploy to Azure:\*\* Use Azure CLI or Azure Portal to deploy the Azure Functions.  
3. \*\*Configure Azure Application Insights:\*\* Set up Application Insights to monitor functions' performance and errors.  
4. \*\*Test the Functions:\*\* Validate each HTTP endpoint using tools like Postman or curl to ensure they behave as expected.  
  
### Conclusion  
  
This generated AIS project includes the necessary structure and code snippets for the Azure Functions that align with the required architecture for integrating with Salesforce. Each function is set up to be modular and follows best practices, ensuring that they can handle their responsibilities independently while maintaining a clean separation of concerns.