Integration Flow

To generate an Azure Integration Services (AIS) project based on the translated architecture and requirements, we need to establish a project structure, configure Azure Functions, and create supporting resources.  
  
### Project Structure  
  
Here’s how the AIS project structure will look:  
  
```  
AIS.Integration  
├── AIS.Integration.Functions  
│ ├── Functions  
│ │ ├── GroupGetByNameFunction.cs  
│ │ ├── ChatterFeedItemPostFunction.cs  
│ │ └── ChatterFeedItemPutFunction.cs  
│ └── local.settings.json  
├── AIS.Integration.Common  
│ ├── Models  
│ │ ├── ErrorDetails.cs  
│ │ └── FeedItem.cs  
│ └── Services  
│ └── SalesforceService.cs  
├── AIS.Integration.Logging  
│ ├── LogService.cs  
│ └── ApplicationInsightsExtensions.cs  
└── AIS.Integration.sln  
```  
  
### Project Files Content  
  
#### 1. \*\*local.settings.json\*\*  
```json  
{  
 "IsEncrypted": false,  
 "Values": {  
 "AzureWebJobsStorage": "UseDevelopmentStorage=true",  
 "FUNCTIONS\_WORKER\_RUNTIME": "dotnet",  
 "SalesforceClientId": "YOUR\_CLIENT\_ID",  
 "SalesforceClientSecret": "YOUR\_CLIENT\_SECRET",  
 "SalesforceUsername": "YOUR\_USERNAME",  
 "SalesforcePassword": "YOUR\_PASSWORD",  
 "SalesforceApiBaseUrl": "https://your.salesforce.api.url"  
 }  
}  
```  
  
#### 2. \*\*GroupGetByNameFunction.cs\*\*  
```csharp  
using System.Net;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Azure.WebJobs;  
using Microsoft.Azure.WebJobs.Extensions.Http;  
using Microsoft.AspNetCore.Http;  
using Microsoft.Extensions.Logging;  
  
namespace AIS.Integration.Functions  
{  
 public static class GroupGetByNameFunction  
 {  
 [FunctionName("GroupGetByName")]  
 public static async Task<IActionResult> Run(  
 [HttpTrigger(AuthorizationLevel.Function, "get", Route = null)] HttpRequest req,  
 ILogger log)  
 {  
 string groupName = req.Query["groupName"];  
 if (string.IsNullOrEmpty(groupName))  
 {  
 return new BadRequestObjectResult("Group name is required.");  
 }  
  
 // Implement logic to fetch Group ID from Salesforce  
 var result = await SalesforceService.FetchGroupIdAsync(groupName);  
 if (result == null)  
 {  
 return new NotFoundResult();  
 }  
  
 return new OkObjectResult(new { Id = result });  
 }  
 }  
}  
```  
  
#### 3. \*\*ChatterFeedItemPostFunction.cs\*\*  
```csharp  
using System.Net;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Azure.WebJobs;  
using Microsoft.Azure.WebJobs.Extensions.Http;  
using Microsoft.AspNetCore.Http;  
using Microsoft.Extensions.Logging;  
  
namespace AIS.Integration.Functions  
{  
 public static class ChatterFeedItemPostFunction  
 {  
 [FunctionName("ChatterFeedItemPost")]  
 public static async Task<IActionResult> Run(  
 [HttpTrigger(AuthorizationLevel.Function, "post", Route = null)] HttpRequest req,  
 ILogger log)  
 {  
 string requestBody = await new StreamReader(req.Body).ReadToEndAsync();  
 var feedItem = JsonConvert.DeserializeObject<FeedItem>(requestBody);  
  
 // Implement logic to post to Chatter  
 var response = await SalesforceService.CreateChatterFeedItemAsync(feedItem);  
 return response ? new OkResult() : new StatusCodeResult((int)HttpStatusCode.InternalServerError);  
 }  
 }  
}  
```  
  
#### 4. \*\*ChatterFeedItemPutFunction.cs\*\*  
```csharp  
using System.Net;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Azure.WebJobs;  
using Microsoft.Azure.WebJobs.Extensions.Http;  
using Microsoft.AspNetCore.Http;  
using Microsoft.Extensions.Logging;  
  
namespace AIS.Integration.Functions  
{  
 public static class ChatterFeedItemPutFunction  
 {  
 [FunctionName("ChatterFeedItemPut")]  
 public static async Task<IActionResult> Run(  
 [HttpTrigger(AuthorizationLevel.Function, "put", Route = null)] HttpRequest req,  
 ILogger log)  
 {  
 string requestBody = await new StreamReader(req.Body).ReadToEndAsync();  
 var feedItem = JsonConvert.DeserializeObject<FeedItem>(requestBody);  
  
 // Implement logic to update Chatter feed item  
 var response = await SalesforceService.UpdateChatterFeedItemAsync(feedItem);  
 return response ? new OkResult() : new StatusCodeResult((int)HttpStatusCode.InternalServerError);  
 }  
 }  
}  
```  
  
#### 5. \*\*ErrorDetails.cs\*\*  
```csharp  
namespace AIS.Integration.Common.Models  
{  
 public class ErrorDetails  
 {  
 public string ProcessName { get; set; }  
 public string SourceSystem { get; set; }  
 public string TargetSystem { get; set; }  
 public string ExceptionType { get; set; }  
 public string EmailSubject { get; set; }  
 public string DefaultResolverGroup { get; set; }  
 public string DefaultEmailCc { get; set; }  
 public string DefaultEmailTo { get; set; }  
 public string DefaultSeverity { get; set; }  
 public string DefaultFail { get; set; }  
 public string IntErrCode { get; set; }  
 }  
}  
```  
  
#### 6. \*\*FeedItem.cs\*\*  
```csharp  
namespace AIS.Integration.Common.Models  
{  
 public class FeedItem  
 {  
 public string Body { get; set; }  
 public string FeedElementType { get; set; }  
 public string SubjectId { get; set; }  
 }  
}  
```  
  
#### 7. \*\*SalesforceService.cs\*\*  
```csharp  
using System.Threading.Tasks;  
using AIS.Integration.Common.Models;  
  
namespace AIS.Integration.Common.Services  
{  
 public static class SalesforceService  
 {  
 public static async Task<string> FetchGroupIdAsync(string groupName)  
 {  
 // Logic to call Salesforce API and retrieve Group ID  
 return "retrieved\_group\_id"; // placeholder  
 }  
  
 public static async Task<bool> CreateChatterFeedItemAsync(FeedItem feedItem)  
 {  
 // Logic to call Salesforce API to create a Chatter feed item  
 return true; // placeholder  
 }  
  
 public static async Task<bool> UpdateChatterFeedItemAsync(FeedItem feedItem)  
 {  
 // Logic to call Salesforce API to update a Chatter feed item  
 return true; // placeholder  
 }  
 }  
}  
```  
  
#### 8. \*\*LogService.cs\*\*  
```csharp  
using Microsoft.Extensions.Logging;  
  
namespace AIS.Integration.Logging  
{  
 public static class LogService  
 {  
 public static void LogError(ILogger log, string message)  
 {  
 log.LogError(message);  
 }  
 }  
}  
```  
  
### Solution Configuration  
- \*\*Create the Solution\*\*: Use Visual Studio to create a new solution named `AIS.Integration.sln`.  
- \*\*Add Projects\*\*: Add the projects `AIS.Integration.Functions` and `AIS.Integration.Common` to the solution, maintaining the structure above and ensuring dependencies are set correctly.  
- \*\*Add Required NuGet Packages\*\*: Add necessary packages like `Microsoft.Azure.WebJobs.Extensions.Http`, `Newtonsoft.Json`, and `Microsoft.Azure.Functions.Extensions`.  
  
### Deployment  
After the code is implemented and thoroughly tested locally:  
- Deploy the Azure Functions to Azure using Azure CLI, Visual Studio, or Azure DevOps.  
- Set up Application Insights for logging and monitoring.  
- Ensure that appropriate security measures are taken when handling sensitive credentials in local.settings.json or Azure Key Vault for production.  
  
### Integration Flow Documentation  
For additional clarity, create an Integration Flow Documentation that describes how data flows through each component, the interactions with Salesforce, and the transformations that take place.  
  
This project should now be aligned with the designed AIS architecture while supporting the original business requirements effectively.