Integration Flow

To create an Azure Integration Services (AIS) project based on the provided architecture and requirements, you can follow these guidelines. The project will consist of the necessary components including Logic Apps, HTTP requests, functions for reusable logic, and observability setup.  
  
### AIS Project Structure  
  
Here's the suggested structure for an AIS project:  
  
```  
/AIS.Integration.Services  
│  
├── /Functions  
│ ├── TransformToGroupId.cs  
│ ├── PrepareChatterMessage.cs  
│ ├── PrepareChatterMessageIfNotExists.cs  
│ └── GenerateAccessToken.cs  
│  
├── /LogicApps  
│ ├── GroupGetByName.json  
│ ├── ChatterFeedItemPost.json  
│ ├── ChatterFeedItemPut.json  
│ └── GenerateChatterAccessToken.json  
│  
├── /Models  
│ ├── GroupGetRequest.cs  
│ ├── ChatterPostRequest.cs  
│ └── AccessTokenRequest.cs  
│  
└── /Observability  
 ├── ApplicationInsightsConfig.cs  
 └── AzureMonitorIntegration.cs  
```  
  
### 1. Create the Azure Functions  
  
#### TransformToGroupId.cs  
```csharp  
using System;  
using Microsoft.AspNetCore.Mvc;  
  
namespace AIS.Integration.Services.Functions  
{  
 public static class TransformToGroupId  
 {  
 /// <summary>  
 /// Transforms the incoming request for group name to retrieve the group ID from Salesforce.  
 /// </summary>  
 [FunctionName("TransformToGroupId")]  
 public static IActionResult Run([HttpTrigger(AuthorizationLevel.Function, "get", Route = "group/{groupName}")] HttpRequest req, string groupName)  
 {  
 // Logic to transform the input and call Salesforce API  
 // Return Group ID or 404 based on the response  
 }  
 }  
}  
```  
  
#### PrepareChatterMessage.cs  
```csharp  
using System;  
  
namespace AIS.Integration.Services.Functions  
{  
 public static class PrepareChatterMessage  
 {  
 /// <summary>  
 /// Prepares the message for posting to Chatter.  
 /// </summary>  
 public static string Run(string originalPayload)  
 {  
 // Logic to prepare Chatter post message  
 }  
 }  
}  
```  
  
#### PrepareChatterMessageIfNotExists.cs  
```csharp  
using System;  
  
namespace AIS.Integration.Services.Functions  
{  
 public static class PrepareChatterMessageIfNotExists  
 {  
 /// <summary>  
 /// Checks if the Chatter post exists and prepares the message if it doesn't.  
 /// </summary>  
 public static string Run(string originalPayload)  
 {  
 // Logic to check for existing Chatter posts  
 }  
 }  
}  
```  
  
#### GenerateAccessToken.cs  
```csharp  
using System;  
  
namespace AIS.Integration.Services.Functions  
{  
 public static class GenerateAccessToken  
 {  
 /// <summary>  
 /// Generates an OAuth access token for Chatter.  
 /// </summary>  
 public static string Run()  
 {  
 // Logic to get access token from Salesforce  
 }  
 }  
}  
```  
  
### 2. Create the Logic Apps  
  
#### GroupGetByName.json  
```json  
{  
 "definition": {  
 "$schema": "http://swagger.schema.url",  
 "triggers": {  
 "manual": {  
 "type": "Request",  
 "inputs": {  
 "method": "GET",  
 "schema": {  
 "type": "object",  
 "properties": {  
 "groupName": {  
 "type": "string"  
 }  
 }  
 }  
 }  
 }  
 },  
 "actions": {  
 "HTTP\_Authorize\_User": {  
 "type": "Http",  
 "inputs": {  
 "method": "GET",  
 "uri": "https://api.example.com/api/v1.0/authorization/authorize"  
 }  
 },  
 "Invoke\_Salesforce\_Connector": {  
 "type": "Http",  
 "inputs": {  
 "method": "GET",  
 "uri": "https://api.salesforce.com/services/data/vXX.X/query?q=SELECT+ID+FROM+CollaborationGroup+WHERE+Name='@{triggerBody()?['groupName']}'"  
 }  
 }  
 }  
 }  
}  
```  
  
#### ChatterFeedItemPost.json  
```json  
{  
 "definition": {  
 "$schema": "http://swagger.schema.url",  
 "triggers": {  
 "manual": {  
 "type": "Request",  
 "inputs": {  
 "method": "POST",  
 "schema": {  
 "type": "object",  
 "properties": {  
 "message": {  
 "type": "string"  
 }  
 }  
 }  
 }  
 }  
 },  
 "actions": {  
 "HTTP\_Authorize\_User": {  
 "type": "Http",  
 "inputs": {  
 "method": "GET",  
 "uri": "https://api.example.com/api/v1.0/authorization/authorize"  
 }  
 },  
 "Post\_To\_Chatter": {  
 "type": "Http",  
 "inputs": {  
 "method": "POST",  
 "uri": "https://api.salesforce.com/services/data/vXX.X/chatter/feed-elements"  
 }  
 }  
 }  
 }  
}  
```  
  
#### ChatterFeedItemPut.json  
```json  
{  
 "definition": {  
 "$schema": "http://swagger.schema.url",  
 "triggers": {  
 "manual": {  
 "type": "Request",  
 "inputs": {  
 "method": "PUT",  
 "schema": {  
 "type": "object",  
 "properties": {  
 "itemId": {  
 "type": "string"  
 }  
 }  
 }  
 }  
 }  
 },  
 "actions": {  
 "HTTP\_Authorize\_User": {  
 "type": "Http",  
 "inputs": {  
 "method": "GET",  
 "uri": "https://api.example.com/api/v1.0/authorization/authorize"  
 }  
 },  
 "Check\_Existing\_Post": {  
 "type": "Http",  
 "inputs": {  
 "method": "GET",  
 "uri": "https://api.salesforce.com/services/data/vXX.X/query?q=SELECT+body+FROM+FeedItem+WHERE+ParentID='@{triggerBody()?['itemId']}'"  
 }  
 },  
 "Post\_To\_Chatter": {  
 "type": "Http",  
 "inputs": {  
 "method": "POST",  
 "uri": "https://api.salesforce.com/services/data/vXX.X/chatter/feed-elements"  
 }  
 }  
 }  
 }  
}  
```  
  
#### GenerateChatterAccessToken.json  
```json  
{  
 "definition": {  
 "$schema": "http://swagger.schema.url",  
 "triggers": {  
 "manual": {  
 "type": "Request",  
 "inputs": {  
 "method": "GET",  
 }  
 }  
 },  
 "actions": {  
 "Get\_Access\_Token": {  
 "type": "Http",  
 "inputs": {  
 "method": "POST",  
 "uri": "https://api.salesforce.com/services/oauth2/token"  
 }  
 }  
 }  
 }  
}  
```  
  
### 3. Models for Payloads  
  
- \*\*GroupGetRequest.cs\*\*  
```csharp  
namespace AIS.Integration.Services.Models  
{  
 public class GroupGetRequest  
 {  
 public string GroupName { get; set; }  
 }  
}  
```  
  
- \*\*ChatterPostRequest.cs\*\*  
```csharp  
namespace AIS.Integration.Services.Models  
{  
 public class ChatterPostRequest  
 {  
 public string Message { get; set; }  
 }  
}  
```  
  
- \*\*AccessTokenRequest.cs\*\*  
```csharp  
namespace AIS.Integration.Services.Models  
{  
 public class AccessTokenRequest  
 {  
 public string ClientID { get; set; }  
 public string ClientSecret { get; set; }  
 public string Username { get; set; }  
 public string Password { get; set; }  
 }  
}  
```  
  
### 4. Configure Observability  
  
#### ApplicationInsightsConfig.cs  
```csharp  
using Microsoft.ApplicationInsights;  
using Microsoft.ApplicationInsights.Extensibility;  
  
namespace AIS.Integration.Services.Observability  
{  
 public static class ApplicationInsightsConfig  
 {  
 public static TelemetryClient TelemetryClient { get; private set; }  
  
 static ApplicationInsightsConfig()  
 {  
 TelemetryConfiguration configuration = TelemetryConfiguration.CreateDefault();  
 configuration.InstrumentationKey = "your-instrumentation-key-here";  
 TelemetryClient = new TelemetryClient(configuration);  
 }  
 }  
}  
```  
  
#### AzureMonitorIntegration.cs  
```csharp  
namespace AIS.Integration.Services.Observability  
{  
 public static class AzureMonitorIntegration  
 {  
 public static void LogEvent(string eventName, IDictionary<string, string> properties)  
 {  
 ApplicationInsightsConfig.TelemetryClient.TrackEvent(eventName, properties);  
 }  
  
 public static void LogException(Exception ex)  
 {  
 ApplicationInsightsConfig.TelemetryClient.TrackException(ex);  
 }  
 }  
}  
```  
  
### Conclusion  
  
This project setup provides a clean structure for implementing the required Azure Integration Services project using Azure Functions, Logic Apps, and integrated observability. Make sure to replace placeholder URIs and keys with actual values from your implementation environment.   
  
You can deploy the project using Azure DevOps or directly from Visual Studio/Azure Portal based on your development preference. Each Logic App can be edited in the Azure portal for any additional approvals or modifications as required by your business logic.