Integration Flow

Below is the integration flow documentation for the project based on Azure Integration Services (AIS) architecture. This documentation outlines the data flow, system interactions, transformations, and external calls in a structured and business-readable format.  
  
---  
  
# AIS Purchase Order Integration Flow Document  
  
## Overview  
This document details the integration flow for the Purchase Order (PO) process leveraging Azure Integration Services (AIS). The flow integrates multiple systems including ERP, Salesforce, and SQL Server to manage purchase orders effectively, automate data retrieval, and ensure data consistency.  
  
## Integration Flow Steps  
  
### 1. \*\*Authorization\*\*  
- \*\*Trigger\*\*: HTTP Trigger  
- \*\*Function\*\*: `POAuthorizationFunction`  
- \*\*Description\*\*: The integration initiates with an authorization request to validate the user's access.  
- \*\*Request Type\*\*: GET  
- \*\*Endpoint\*\*: `/authorize`  
- \*\*Response\*\*: Confirms whether the user is authorized.  
  
### 2. \*\*Get Last Run Timestamp\*\*  
- \*\*Function\*\*: `GetLastRunTimestampFunction`  
- \*\*Description\*\*: Retrieves the timestamp of the last executed Purchase Order integration process to ensure only new or modified records are processed.  
- \*\*Request Type\*\*: GET  
- \*\*Endpoint\*\*: SQL Server API - `SchedulerTimestamp`  
- \*\*Output\*\*: LastRun.Timestamp (date-time)  
  
### 3. \*\*PO Receipt Data Retrieval\*\*  
- \*\*Function\*\*: `POReceiptQueryFunction`  
- \*\*Description\*\*: Queries Salesforce to retrieve all new purchase order receipts since the LastRun timestamp.  
- \*\*Request Type\*\*: Salesforce Query  
- \*\*Query\*\*: SQL-like query to fetch PO receipt data.  
- \*\*Output\*\*: List of `POReceiptData` objects.  
  
### 4. \*\*Distinct Purchase Orders\*\*  
- \*\*Transformation\*\*: Identify distinct purchase orders from the retrieved receipt data.  
- \*\*Logic\*\*: Filter and map distinct Purchase Order IDs based on the receipt data.  
- \*\*Output\*\*: List of `DistinctPOs` (unique purchase orders).  
  
### 5. \*\*Purchase Order Data Retrieval\*\*  
- \*\*Function\*\*: `POQueryFunction`  
- \*\*Description\*\*: Queries Salesforce to retrieve detailed information for each distinct purchase order.  
- \*\*Request Type\*\*: Salesforce Query  
- \*\*Query\*\*: SQL-like query to fetch detailed purchase order data based on `DistinctPOs`.  
- \*\*Output\*\*: List of `POData` objects.  
  
### 6. \*\*Put PO Data to ERP\*\*  
- \*\*Function\*\*: `PutPOtoERPFunction`  
- \*\*Description\*\*: Sends the processed purchase order data to the ERP system for further processing or record-keeping.  
- \*\*Request Type\*\*: PUT  
- \*\*Endpoint\*\*: ERP API - `/api/v1.0/PurchaseOrder`  
- \*\*Body\*\*: Transformed `POData` for ERP consumption.  
  
### 7. \*\*Update Last Run Timestamp\*\*  
- \*\*Function\*\*: `PostLastRunTimestampFunction`  
- \*\*Description\*\*: Updates the SQL Server with the latest run timestamp for tracking purposes.  
- \*\*Request Type\*\*: POST  
- \*\*Endpoint\*\*: SQL Server API - `SchedulerTimestamp`  
- \*\*Body\*\*: JSON object containing updated LastRunTimestamp.  
  
### 8. \*\*Get Company References\*\*  
- \*\*Function\*\*: `CompanyReferenceFunction`  
- \*\*Description\*\*: Queries Salesforce for company references that can be used within the integration.  
- \*\*Request Type\*\*: Salesforce Query  
- \*\*Output\*\*: List of company reference objects.  
  
### 9. \*\*Vendor Address Query\*\*  
- \*\*Function\*\*: `VendorAddressQueryFunction`  
- \*\*Description\*\*: Retrieves vendor address information based on the purchase order data.  
- \*\*Request Type\*\*: Salesforce Query for each PO record.  
- \*\*Output\*\*: Enhanced `POData` that includes `VendorAddressNumber`.  
  
## Data Transformations and Mappings  
- Transformation logic is implemented within each respective Azure Function to convert and structure data as it flows through the system.  
- Use of DataWeave-style transformations enables seamless data preparation for queries and API calls.  
  
## Error Handling and Logging  
- Error handling strategies are implemented in all functions to catch and manage exceptions effectively.  
- All actions, including success and error scenarios, are logged using Application Insights for monitoring and auditing purposes.  
  
## Conclusion  
This integration flow encapsulates the process of retrieving, processing, and sending purchase order data between multiple systems. The modular design allows for scaling, maintenance, and adaptability to changing business requirements. By utilizing Azure Integration Services, the organization can achieve a high degree of automation and efficiency in managing purchase orders.  
  
---  
  
This integration flow document serves as a comprehensive guide for developers and stakeholders to understand the architecture, data flow, and system interactions within the Purchase Order integration project. Further details can be added as the project evolves or as specific requirements are identified.