Integration Flow

# Flow Breakdown

## Step Below is the \*\*Integration Flow Document\*\* for the "Purchase Order Integration" process:

---

# \*\*Purchase Order Integration Flow\*\*

### \*\*Objective\*\*

Facilitate integration between Salesforce (source system) and downstream systems (ERP APIs). This integration fetches Purchase Order (PO) data and submits it to an external ERP system while ensuring robust error handling, data transformation, and observability.

---

## \*\*Integration Flow Overview\*\*

| \*\*Source System\*\* | \*\*Processing Logic\*\* | \*\*Target System(s)\*\* |

|-----------------------|------------------------------------------------------------------------------------------------------|-----------------------|

| \*\*Salesforce\*\* | - Query and retrieve Purchase Order data and related entities (PO Lines, Receipts, Vendor details). | External ERP (e.g. Rootstock) |

| \*\*Azure Service Bus\*\* | - Publish integration events for downstream system notifications. | ERP Workflow Process |

| \*\*HTTP Trigger\*\* | - Expose an API endpoint for real-time initiation of the process. | |

---

### \*\*Assumptions\*\*

1. Data transformation rules adhere to Salesforce-specific field mappings and downstream system requirements.

2. Service Bus topics dynamically handle event subscription needs.

3. Observability design tracks the integration for failures, retries, and alerts.

---

## \*\*Flow Breakdown\*\*

### \*\*1. Initial Trigger\*\*

**- \*\*Trigger Type:** \*\* HTTP Trigger (real-time) or Timer Trigger (scheduled runs).

**- \*\*Purpose:** \*\* Starts the integration workflow.

**- \*\*System Interaction:** \*\*

- Reads \*\*last-run timestamp\*\* for incremental processing.

- Invokes \*\*Salesforce API\*\* to retrieve process-relevant PO data.

---

### \*\*2. Retrieve PO Information\*\*

- \*\*

## Step 1: Query Purchase Orders\*\*

**- \*\*Action:** \*\* Fetch Purchase Order data from Salesforce where `LastModifiedDate > LastRunTimestamp`.

**- \*\*Query Sample:** \*\*

```sql

SELECT ID, Name, LastModifiedDate, Status, Vendor, BillingAddress, ShippingAddress

FROM PurchaseOrder

WHERE LastModifiedDate > '<LastRunTimestamp>'

```

**- \*\*System:** \*\* Salesforce.

**- \*\*Output Variables:** \*\* `PurchaseOrders`.

- \*\*

## Step 2: Retrieve PO Line Items\*\*

**- \*\*Action:** \*\* For each Purchase Order ID, fetch related Line Items from Salesforce.

**- \*\*Query Sample:** \*\*

```sql

SELECT ID, Name, QuantityRequested, Description

FROM POLineItem

WHERE PurchaseOrderID IN ('<DistinctPOIDs>')

```

**- \*\*System:** \*\* Salesforce.

**- \*\*Output Variables:** \*\* `POLineItems`.

- \*\*

## Step 3: Retrieve Vendor Information\*\*

**- \*\*Action:** \*\* Query vendor addresses based on the PO’s billing address postal code and vendor external system mapping.

**- \*\*Query Sample:** \*\*

```sql

SELECT ID, AddressNumber

FROM VendorAddress

WHERE VendorCode = '<PO.VendorCode>'

AND PostalCode = '<PO.BillingAddress.PostalCode>'

```

**- \*\*System:** \*\* Salesforce.

**- \*\*Output Variables:** \*\* `VendorAddressNumber`.

---

### \*\*3. Data Transformation\*\*

**- \*\*Purpose:** \*\* Map Salesforce data to the downstream ERP API schema.

**- \*\*Transformation Logic:** \*\*

- Combine results from `PurchaseOrders`, `POLineItems`, and `VendorAddressNumber` to build the API request payload.

**- Perform filtering:** Exclude Purchase Orders with inactive statuses.

**- \*\*Sample PO Data Model:** \*\*

```json

{

**"PurchaseOrderNumber":** "PO12345",

**"Status":** "Active",

**"VendorCode":** "V123",

**"BillingAddress":** {...},

**"ShippingAddress":** {...},

**"LineItems":** [

{

**"LineNumber":** 1,

**"Description":** "Widgets",

**"Quantity":** 100,

**"UnitPrice":** 10.50

}

],

**"VendorAddressNumber":** "A123"

}

```

---

### \*\*4. Push Data to ERP\*\*

**- \*\*Action:** \*\* Submit PO data to the ERP system API endpoint.

**- \*\*Target API Endpoint:** \*\* `/api/v1.0/PurchaseOrder`.

**- \*\*Request Example:** \*\*

```json

{

**"POData":** [

{

**"PurchaseOrderNumber":** "PO12345",

**"VendorCode":** "V123",

**"BillingAddress":** {...},

**"ShippingAddress":** {...},

**"LineItems":** [...]

}

]

}

```

**- \*\*System:** \*\* Rootstock ERP or similar external system.

---

### \*\*5. Post-Processing\*\*

**- \*\*Action 1:** \*\* Update the \*\*Last Run Timestamp\*\* for future incremental runs.

**- \*\*Target:** \*\* `/SchedulerTimestamp`.

**- \*\*Payload:** \*\*

```json

{

**"Process":** "PurchaseOrder",

**"LastRunTimestamp":** "<current-time>"

}

```

**- \*\*Action 2:** \*\* Publish an event to \*\*Azure Service Bus\*\*.

**- \*\*Topic:** \*\* `IntegrationEvents`.

**- \*\*Event Payload:** \*\*

```json

{

**"EventType":** "PurchaseOrderProcessed",

**"Timestamp":** "<current-timestamp>",

**"Details":** {

**"PurchaseOrderNumber":** "PO12345"

}

}

```

---

### \*\*6. Exception Handling\*\*

**Defined error handling strategies across all modules:**

**1. \*\*HTTP Retry Mechanism\*\*:** Retry on transient errors.

**2. \*\*Service Bus Dead Letter Queue (DLQ):** \*\* Capture unprocessable messages.

**3. \*\*Global Exception Strategy:** \*\* Centralized exception logging with Application Insights telemetry.

---

## \*\*High-Level Flow Diagram\*\*

```

[ HTTP Trigger ] --> [ Retrieve LastRunTimestamp ] --> [ Query PO Data ]

|

v

[ Query PO Line Items ]

|

v

[ Query Vendor Address Info ]

|

v

[ Transform Data for ERP Submission ] -----> [ Push Data to ERP ]

|

v

[ Update Scheduler Timestamp ] -----> [ Publish Event to Service Bus ]

```

---

## \*\*Flow Breakdown (Steps)\*\*

### \*\*

## Step 1: Start Integration\*\*

- Triggered via HTTP request with optional parameters (e.g., force re-run with a specific date).

- Fetch last-run timestamp using SQL integration API call.

---

### \*\*

## Step 2: Querying PO Data\*\*

- Main query retrieves all relevant Purchase Orders modified since the last run.

- Loops through distinct Purchase Order IDs to fetch related line items.

---

### \*\*

## Step 3: Enrich Data\*\*

- Fetch vendor addresses based on billing addresses.

- Combine and transform for ERP requirements.

---

### \*\*

## Step 4: Push Data\*\*

- Use structured payload and push final data to downstream ERP.

---

### \*\*

## Step 5: Publish Event & Update Timestamp\*\*

- Submit success/failure events to \*\*Azure Service Bus\*\* for downstream processing.

- Update the timestamp API for tracking the last processed transaction.

---

## \*\*Error Scenarios\*\*

**### Retryable Errors:**

- Transient network/API failures (e.g., HTTP Timeouts) are retried with exponential backoff.

**### Non-Retryable Errors:**

- Data validation issues (e.g., malformed payloads) are logged and ignored after DLQ capture.

**### Logging and Monitoring:**

- All errors and telemetry are logged in \*\*Application Insights\*\* for centralized observability.

---

This Integration Flow Document aligns with the AIS architecture while ensuring clarity, modularity, and scalability. Let me know if updates or diagrams are needed!