

Phase 7 — Integration & External Access (Demo / Mock Implementations)

Project: Revolutionizing Agriculture with AgriEdge Or-Mange Ltd — Salesforce OMS

Prepared for: Sai Teja Kotipalli

Status: Completed (integration components implemented as **simulations / mock callouts** suitable for a Developer Org; production connections require external API credentials and environment)

1. Executive summary

This document lists everything created for **Phase 7: Integration & External Access**, describes the purpose of each artifact, where to find it in your org, how it was implemented (mock/demo mode), and how to verify and test it.

Important: Because a Developer Edition org does not typically have live external service credentials, the integrations were implemented as **mock / simulated** callouts and platform events to demonstrate the design and end-to-end flow. The document notes what to change for a production-ready integration.

2. Summary of artifacts created (what was implemented)

A. Named Credentials (mock endpoints)

- **MockPaymentGateway**
 - Purpose: Named credential for simulated payment gateway callouts.
 - Setup path: Setup → Security → Named Credentials → MockPaymentGateway
 - URL used (demo): <https://mock-payment-gateway.com>

The screenshot shows the Salesforce 'Named Credentials' configuration page for 'MockPaymentGateway'. The page is titled 'SETUP > NAMED CREDENTIALS' and 'MockPaymentGateway'. It includes fields for 'Label' (MockPaymentGateway), 'Name' (MockPaymentGateway), and 'URL' (https://mock-payment-gateway.com/api). There is a toggle for 'Enabled for Callouts' which is checked. Below this are sections for 'Authentication' (External Credential: MockPaymentAuth, Client Certificate) and 'Callout Options' (Generate Authorization Header: checked, Allow Formulas in HTTP Header: unchecked, Allow Formulas in HTTP Body: unchecked, Outbound Network Connection: 1).

- **MockLogisticsAPI**
 - Purpose: Named credential for simulated logistics partner callouts.

- Setup path: Setup → Security → Named Credentials → MockLogisticsAPI
- URL used (demo): <https://mock-logistics.com/api>

The screenshot shows the 'MockLogisticsAPI' Named Credential configuration in Salesforce Setup. The breadcrumb trail is 'SETUP > NAMED CREDENTIALS'. The configuration includes:

- Label:** MockLogisticsAPI
- Name:** MockLogisticsAPI
- URL:** <https://mock-logistics.com/api>
- Enabled for Callouts:** Checked (toggle switch)
- Authentication:**
 - External Credential:** MockPaymentAuth
 - Client Certificate:** (empty field)
- Callout Options:**
 - Generate Authorization Header:** Checked
 - Allow Formulas in HTTP Header:** Unchecked
 - Allow Formulas in HTTP Body:** Unchecked
 - Outbound Network Connection:** (empty field)

These are placeholders so Apex callouts use a central credential. Replace with real URLs and auth (OAuth, certificate, or API key) when moving to sandbox/production.

B. Apex Callout / Service Classes (mock implementations)

- **PaymentGatewayService** (Apex Class)
 - Purpose: Demonstrates how the app would call a payment gateway to process online payments.
 - Demo behavior: Logs a simulated success and returns 'Success'.

```

1 public with sharing class PaymentGatewayService {
2     public static String processPayment(String orderId, Decimal amount){
3         // Mock callout
4         System.debug('Payment processed for Order: ' + orderId + ', Amount: ' + amount);
5         return 'Success';
6     }
7 }
  
```

- **LogisticsService** (Apex Class)

- Purpose: Simulated REST callout to logistics partner to fetch/update delivery status. Implemented as a future method to allow callouts asynchronously.
- Demo behavior: Updates Delivery__c.Status__c to a simulated value (e.g., Dispatched) and logs the operation.

```

1 public with sharing class LogisticsService {
2     @future(callout=true)
3     public static void updateDeliveryStatus(String deliveryId){
4         System.debug('Simulating API call to logistics for Delivery: ' + deliveryId);
5         // Mock response
6         String status = 'Dispatched';
7         Delivery__c del = [SELECT Id, Status__c FROM Delivery__c WHERE Id=:deliveryId LIMIT 1];
8         del.Status__c = status;
9         update del;
10    }
11 }

```

- **NotificationService (Apex Class)**

- Purpose: Asynchronous notifications (SMS/WhatsApp/email) via @future methods. For demo, logs message content. For production, replace contents with HTTP callouts to Twilio/WhatsApp API or use middleware.

```

1 public class NotificationService {
2     @future
3     public static void sendSMS(String phone, String message){
4         System.debug('SMS/WhatsApp sent to: ' + phone + ', Message: ' + message);
5     }
6 }

```

C. Platform Event

- **OrderUpdateEvent__e (Platform Event)**

- Fields: OrderId__c (Text), Status__c (Text)
- Purpose: Publish events when Order status changes so multiple subscribers (Flows, external listeners) can react in near real-time.
- Setup path: Setup → Platform Events → OrderUpdateEvent
- Publisher: FireOrderEvent trigger (see below) publishes events when an Order status changes.

Platform Events

Platform Event
OrderUpdateEvent

[Standard Fields \(5\)](#) | [Custom Fields & Relationships \(2\)](#)

Platform Event Definition Detail

Edit Delete

Singular Label	OrderUpdateEvent	Description	
Plural Label	OrderUpdateEvents	Deployment Status	Deployed
Object Name	OrderUpdateEvent		
API Name	OrderUpdateEvent__e		
Event Type	High Volume i		
Publish Behavior	Publish Immediately i		
Created By	Kotipalli Sai Teja, 9/25/2025, 11:10 AM	Modified By	Kotipalli Sai Teja, 9/25/2025, 11:10 AM

Standard Fields

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
	Created By	CreatedBy	Lookup(User)		
	Created Date	CreatedDate	Date/Time		
	Event UUID	EventUuid	Text(36)		
	Replay ID	ReplayId	External Lookup		

Custom Fields & Relationships

New

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit Del	OrderId	OrderId__c	Text(18)			Kotipalli Sai Teja, 9/25/2025, 11:11 AM
Edit Del	Status	Status__c	Text(255)			Kotipalli Sai Teja, 9/25/2025, 11:12 AM

Triggers

New

D. Apex Trigger to Publish Platform Event

- FireOrderEvent** (Trigger on Order__c)
 - Purpose: On after update, if Order_Status__c changed, publish OrderUpdateEvent__e with new status.
 - Where to find: Setup → Custom Code → Apex Triggers → FireOrderEvent

Developer Console - Google Chrome

orgfarm-4c5183c8d1-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

InventoryService.apxc
OrderItemTrigger.apxt
DeliveryTrigger.apxt
StockReconciliationBatch.apxc
InventoryServiceTest.apxc
PaymentGatewayService.apxc
NotificationService.apxc
LogisticsService.apxc
FireOrderEvent.apxt

Code Coverage: None | API Version: 64

```

1 trigger FireOrderEvent on Order1__c (after update) {
2     List<OrderUpdateEvent__e> events = new List<OrderUpdateEvent__e>();
3     for(Order1__c ord : Trigger.new){
4         if(ord.Order_Status__c != Trigger.oldMap.get(ord.Id).Order_Status__c){
5             events.add(new OrderUpdateEvent__e(OrderId__c=ord.Id, Status__c=ord.Order_Status__c));
6         }
7     }
8     if(events.size() > 0){
9         EventBus.publish(events);
10    }
11 }

```