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

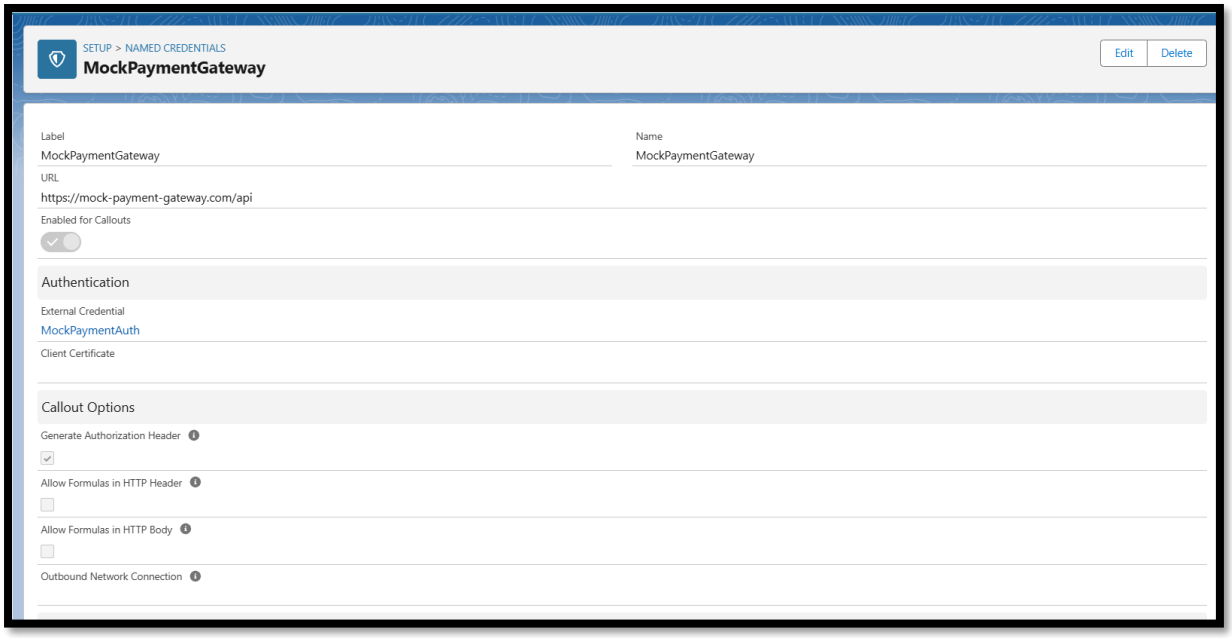
This document lists everything created for **Phase 7: Integration & External Access**, describes the purpose of each artifact, where to find it in your org, how it was implemented (mock/demo mode), and how to verify and test it.
Important: Because a Developer Edition org does not typically have live 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)

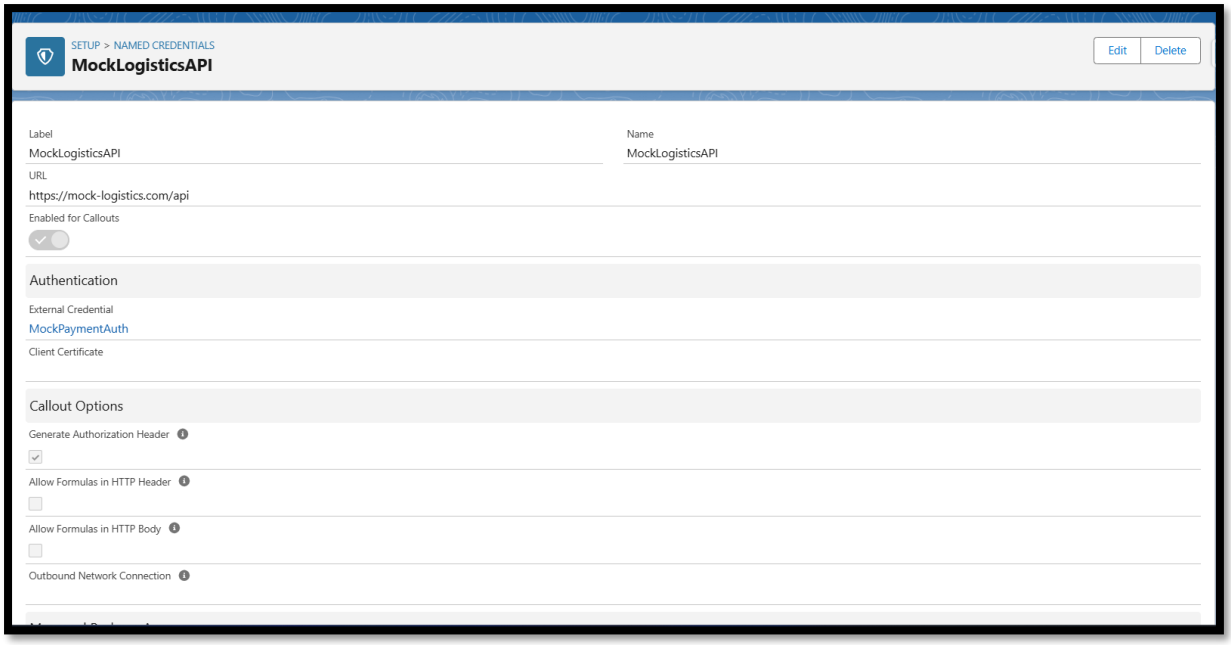
A. Named Credentials (mock endpoints)

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



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

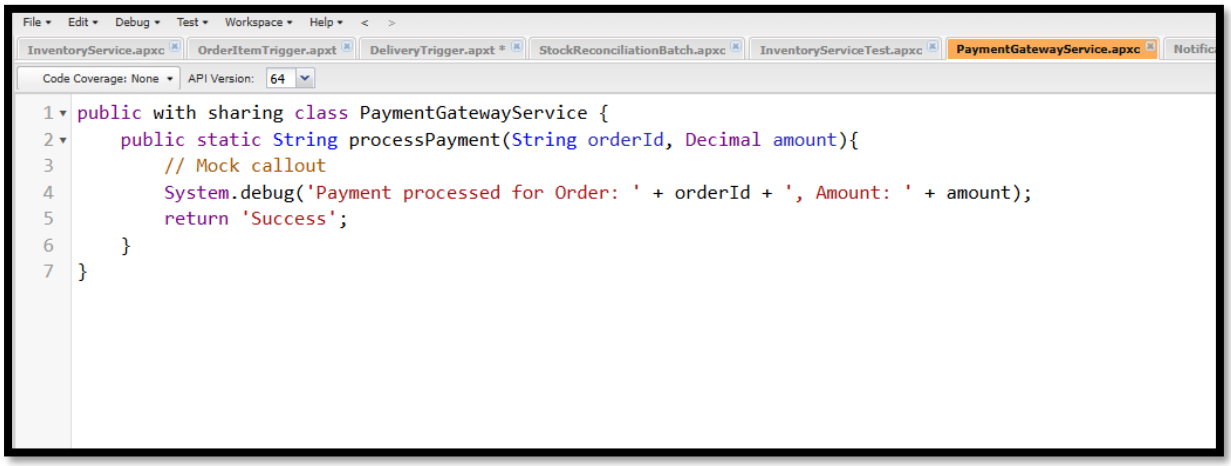
- Setup path: Setup → Security → Named Credentials → MockLogisticsAPI
- URL used (demo): https://mock-logistics.com/api



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'.



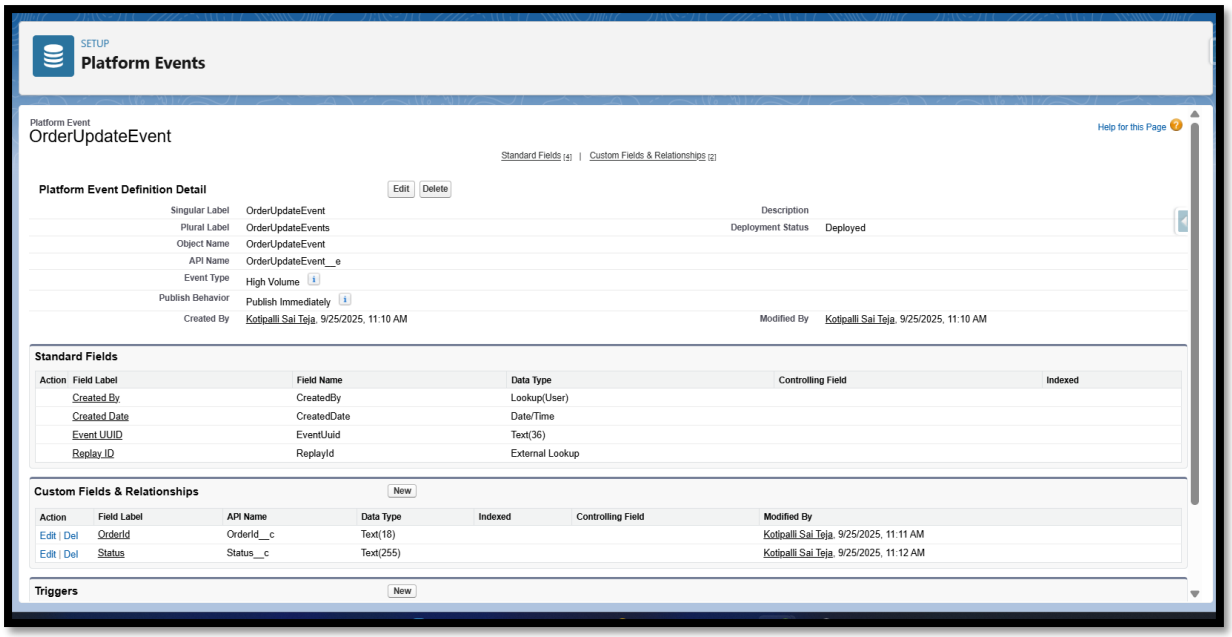
- **LogisticsService** (Apex Class)

- 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.

- **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.

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.



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