

## *Phase 7: Integration & External Access*

### **Objective:**

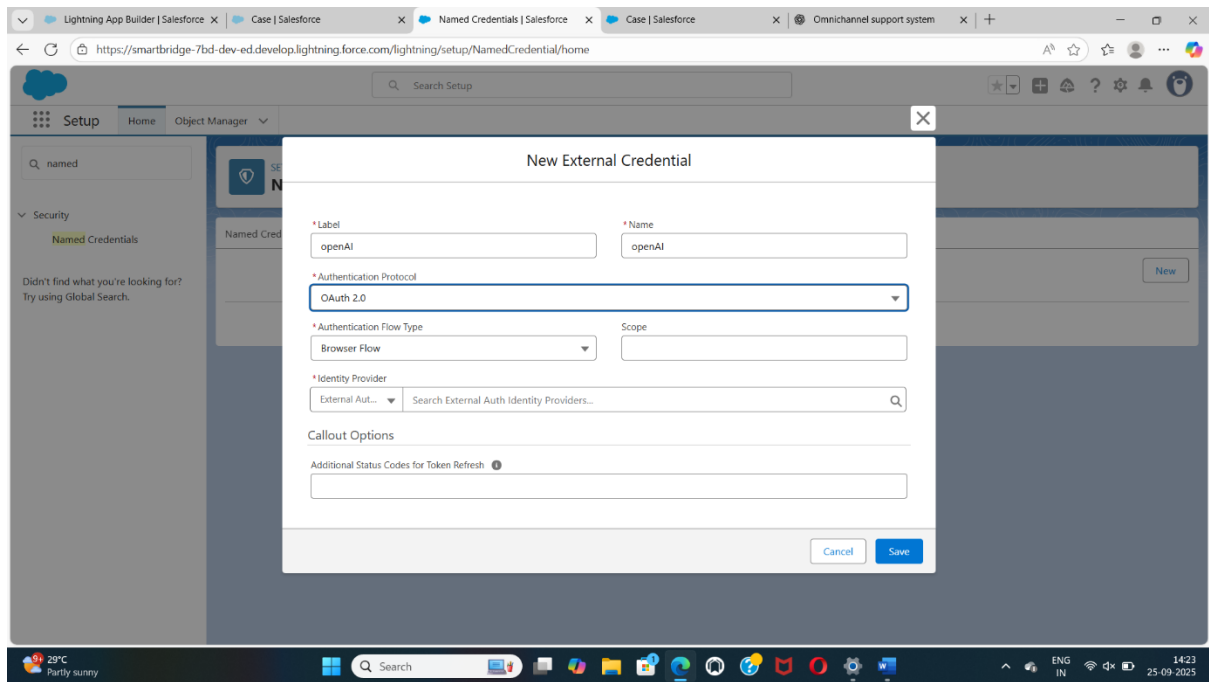
Enable the Salesforce org to securely communicate with external systems (WhatsApp, social media, or other applications) so that customer queries, notifications, and external data can flow seamlessly into Salesforce and vice versa.

### **1. Named Credentials**

- **Purpose:** Store and manage authentication details for external systems easily.
- **Example:** Twilio for WhatsApp integration.
- **Implementation Steps:**
  1. Go to **Setup → Named Credentials → New Named Credential**.
  2. Enter:
    - Label & Name
    - URL of external API (e.g., <https://api.twilio.com>)
    - Authentication method (OAuth 2.0 / Password)
  3. Save.
- **Outcome:** Apex classes can call external APIs without hardcoding credentials.

### **2. External Services**

- **Purpose:** Connect Salesforce to external APIs without complex code.
- **Example:** Calling a social media API to fetch customer messages.
- **Implementation Steps:**
  1. Go to **Setup → External Services → New External Service**.
  2. Provide the API specification (Swagger/OpenAPI).
  3. Salesforce generates invocable actions that can be used in Flows or Apex.
- **Outcome:** Simplifies integration and enables automation using external data.



### 3. Web Services (REST/SOAP)

- **Purpose:** Enable Salesforce to send or receive data from external systems.
- **Implementation Steps:**
  1. Enable API access for users/profiles.
  2. Use **Apex HTTP Callouts** or **External Services**.
  3. Test integration using Postman or Workbench.
- **Example in Project:**
  - WhatsApp messages → REST API call to create Cases.
  - Salesforce notifications → REST API call to external apps.

### 4. Apex Callouts

- **Purpose:** Connect Salesforce with external APIs via Apex code.
- **Implementation Steps:**
  1. Create Apex class for callout (e.g., WhatsAppNotifier).
  2. Use **HTTP methods (GET/POST)** to send/receive data.
  3. Call this class from **Triggers or Flows**.

- **Important Notes:**

- Use @future or Queueable Apex for long-running callouts.
- Ensure **Remote Site Settings** or Named Credentials are configured.

## 5. Platform Events

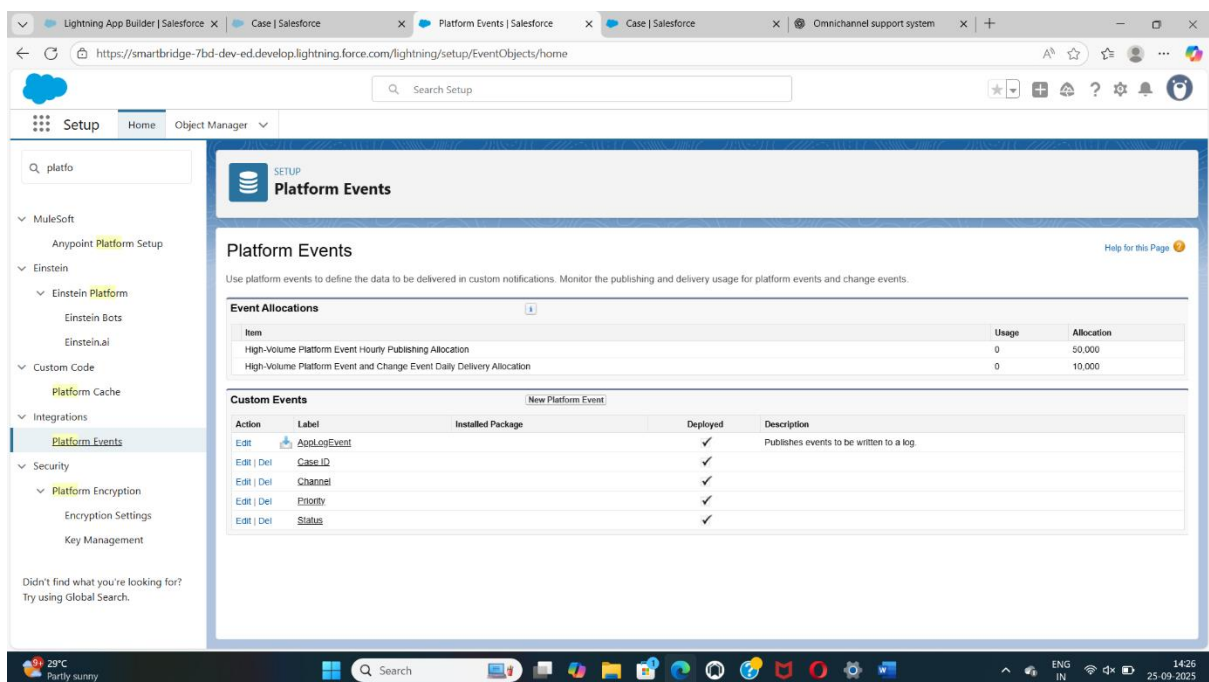
- **Purpose:** Real-time communication between Salesforce and external systems.

- **Implementation Steps:**

1. Setup → **Platform Events** → **New Platform Event**.
2. Define fields (Case ID, Priority, Channel, Status).
3. Subscribe via **Apex Trigger** or external system.

- **Example in Project:**

- When a Case is created → Platform Event triggers → external dashboard updates instantly.



The screenshot shows the Salesforce Setup interface for Platform Events. The left sidebar contains navigation options like MuleSoft, Einstein, Custom Code, Integrations, and Security. The main content area is titled 'Platform Events' and includes a description: 'Use platform events to define the data to be delivered in custom notifications. Monitor the publishing and delivery usage for platform events and change events.'

Below the description, there are two tables:

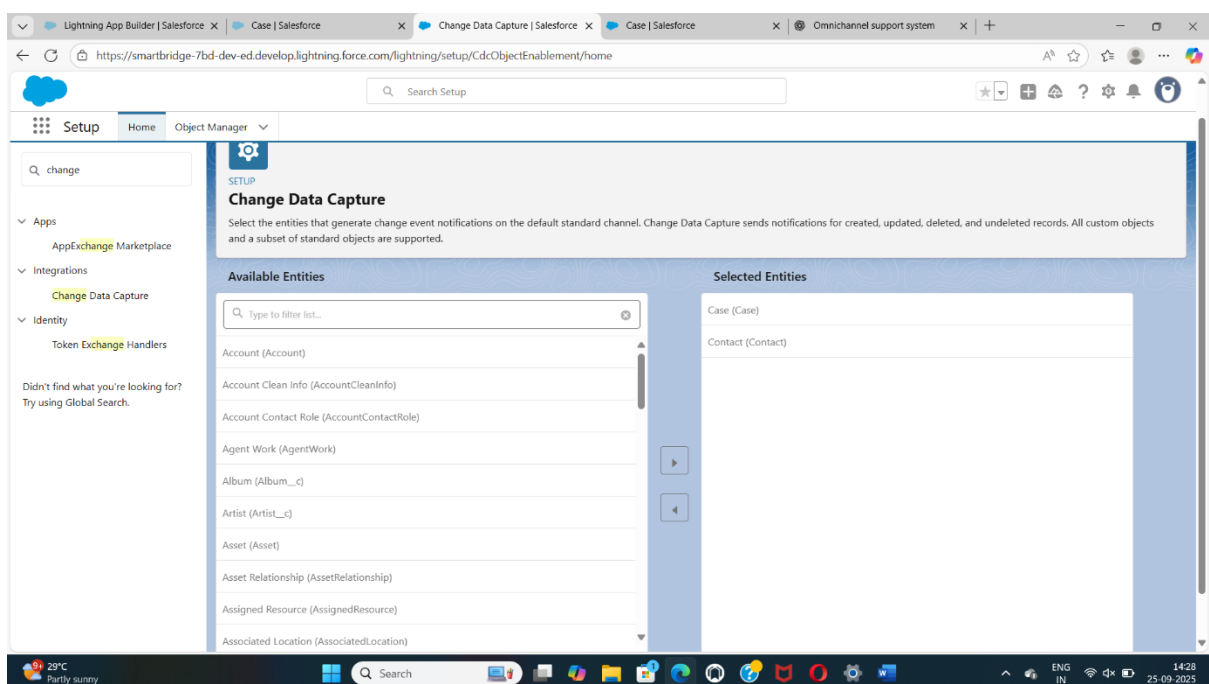
Event Allocations		
Item	Usage	Allocation
High-Volume Platform Event Hourly Publishing Allocation	0	50,000
High-Volume Platform Event and Change Event Daily Delivery Allocation	0	10,000

Custom Events				
Action	Label	Installed Package	Deployed	Description
Edit	ApexLogEvent		✓	Publishes events to be written to a log.
Edit   Del	Case ID		✓	
Edit   Del	Channel		✓	
Edit   Del	Priority		✓	
Edit   Del	Status		✓	

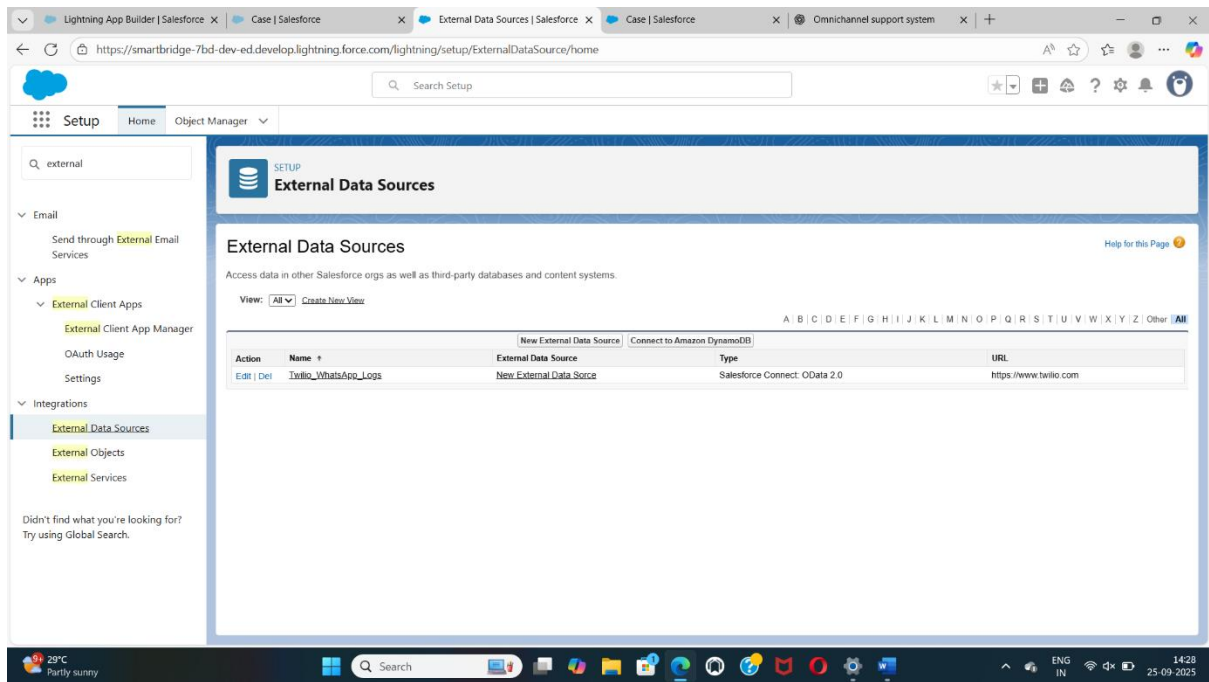
## 6. Change Data Capture

- **Purpose:** Notify external systems when Salesforce records are created, updated, or deleted.
- **Implementation Steps:**
  1. Setup → **Change Data Capture** → **Enable for objects** (e.g., Case, Contact).
  2. Subscribe via **CometD** or **Streaming API**.
- **Example in Project:**
  - Case status changes to “Resolved” → external analytics system updates automatically.



## 7. Salesforce Connect (External Objects)

- **Purpose:** Access external database records without storing them in Salesforce.
- **Implementation Steps:**
  1. Setup → **External Data Sources** → **New External Data Source**.
  2. Select connector type (OData 2.0 / 4.0).
  3. Create **External Object** to display external records in Salesforce UI.
- **Example in Project:**
  - WhatsApp chat logs stored externally → visible to agents inside Salesforce.

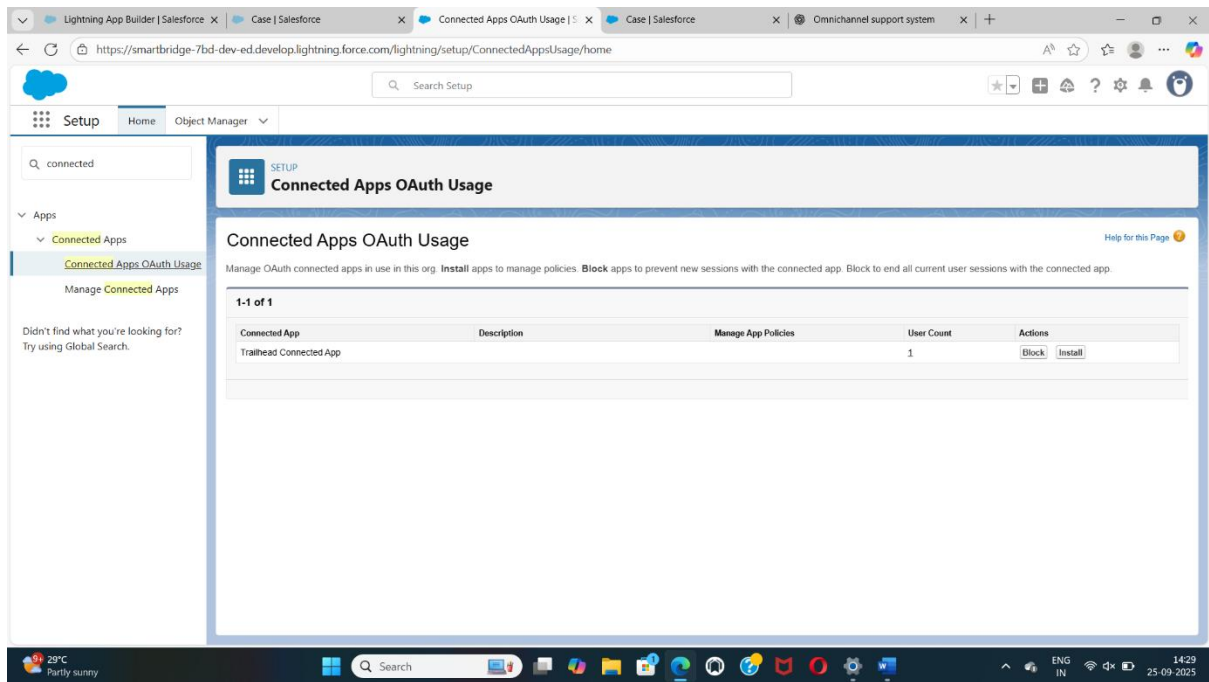


## 8. API Limits

- **Purpose:** Salesforce enforces limits on API calls to prevent system overload.
- **Example:** Developer edition allows 15,000 API calls/day.
- **Tip:** Optimize callouts, batch requests, or use Platform Events to reduce API usage.

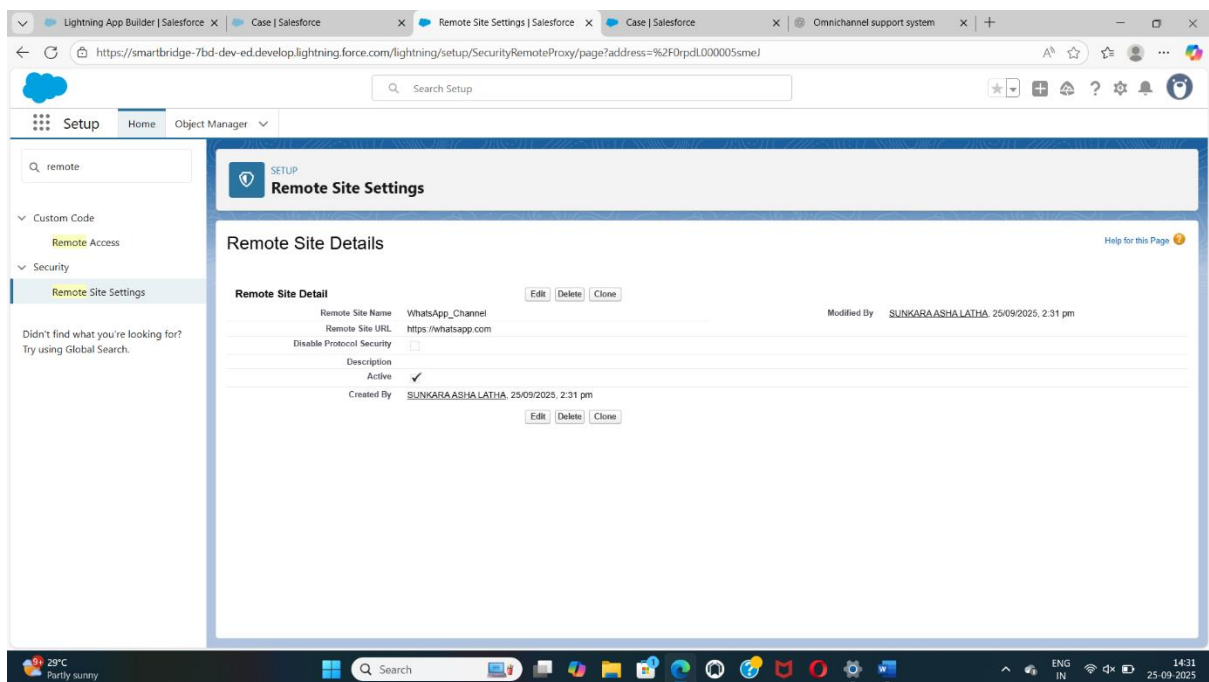
## 9. OAuth & Authentication

- **Purpose:** Securely connect external applications with Salesforce.
- **Implementation Steps:**
  1. Setup → **Connected App** → **New**.
  2. Configure OAuth scopes (e.g., full, api).
  3. Use **Access Tokens** in Apex/Flows instead of username/password.



## 10. Remote Site Settings

- **Purpose:** Required for Apex callouts to URLs not covered by Named Credentials.
- **Implementation Steps:**
  1. Setup → Remote Site Settings → New.
  2. Add external endpoint URL.



### **Expected Outputs / Results**

- Cases automatically created from WhatsApp or social media messages.
- External APIs called successfully via Apex or Flows.
- Platform Events notify external systems in real-time.
- Change Data Capture ensures updates are pushed automatically.
- Agents can view external WhatsApp logs inside Salesforce.
- OAuth authentication ensures secure API access.