

# Phase 7 Report

## Integration & External Access

**Project:** Smart Healthcare Appointment & Compliance Hub

**Batch:** 4

**Program:** TCS Last Mile SmartBridge

**Prepared by:** Palla Bhugarbha

## 1. Introduction

---

This phase focuses on enabling secure integration and external access for the Smart Healthcare Appointment & Compliance Hub. By configuring Named Credentials, Remote Site Settings, and Apex callouts, the system can communicate with external healthcare APIs such as lab systems, billing services, or compliance databases. In addition, real-time integration mechanisms like Platform Events ensure that critical updates, such as new appointments or compliance expiries, are immediately shared with external systems. These integrations extend the functionality of the application beyond Salesforce, ensuring interoperability with hospital infrastructure and external services.

## 2. Objectives

---

- Configure **Named Credentials** and **External Credentials** to securely store authentication details and external endpoints.
- Implement **Apex callouts** to exchange appointment and compliance data with external healthcare APIs.
- Set up **Platform Events** to notify external systems in real time when appointments or compliance changes occur.
- Ensure secure access by applying **Remote Site Settings** and respecting Salesforce **API limits**.
- Demonstrate end-to-end integration with test callouts and event monitoring to validate external access.

## 3. Steps Performed

---

### 3.1 Named Credentials + Callouts

In the healthcare hub, the system needs to **send appointment details** (patient name, clinician, time) to an **external lab/billing API**. Instead of hardcoding credentials in Apex, we create a **Named Credential** and then write Apex code to do a secure REST API callout.

### 3.1.1 Creating Named Credential

In this step, the Smart Healthcare Hub demonstrates **secure external integration**. When appointment details need to be shared with an external system—such as a lab or billing service—the platform uses **Named Credentials** to store connection information securely. This avoids hardcoding sensitive credentials in Apex code and simplifies maintenance. The **Apex callout** then uses the Named Credential to send appointment information (patient name, clinician, and appointment time) to the external API in real time. This ensures data consistency across internal and external systems while maintaining security and compliance. By using a sample API for testing, the workflow proves that Salesforce can communicate with external services reliably.

The screenshot shows the 'New Named Credential' dialog box in the Salesforce Setup interface. The dialog is titled 'New Named Credential' and contains the following fields and options:

- Label:** ExternalLabAPI
- Name:** ExternalLabAPI
- URL:** https://jsonplaceholder.typicode.com
- Enabled for Callouts:** ☒
- Authentication:**
  - External Credential:** PublicAPICredential (selected from a dropdown menu)
  - Client Certificate:** Search Certificates...
- Callout Options:**
  - Generate Authorization Header:** ☒

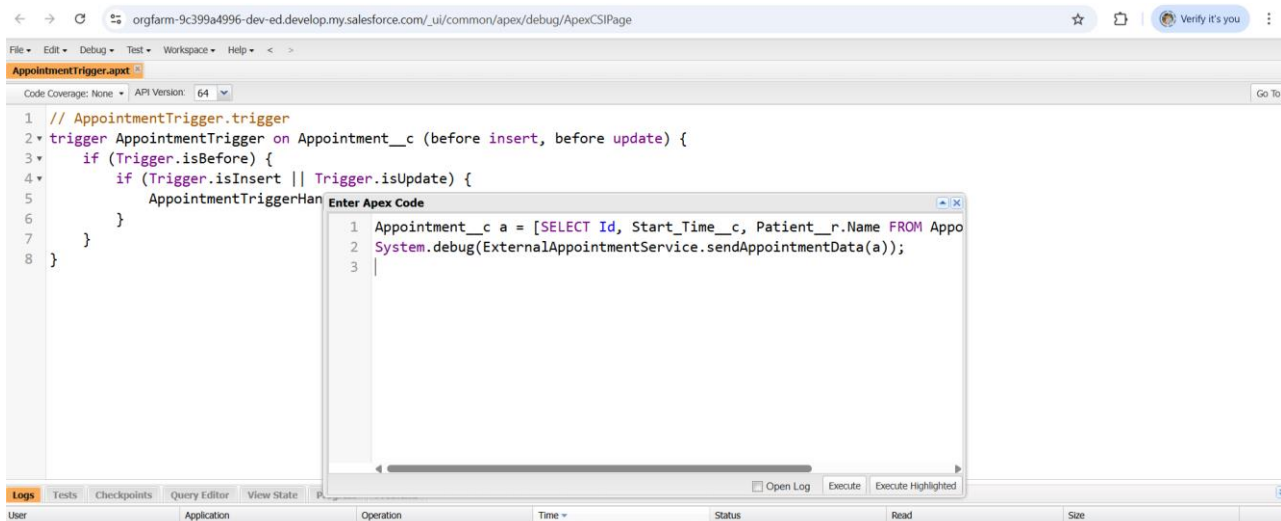
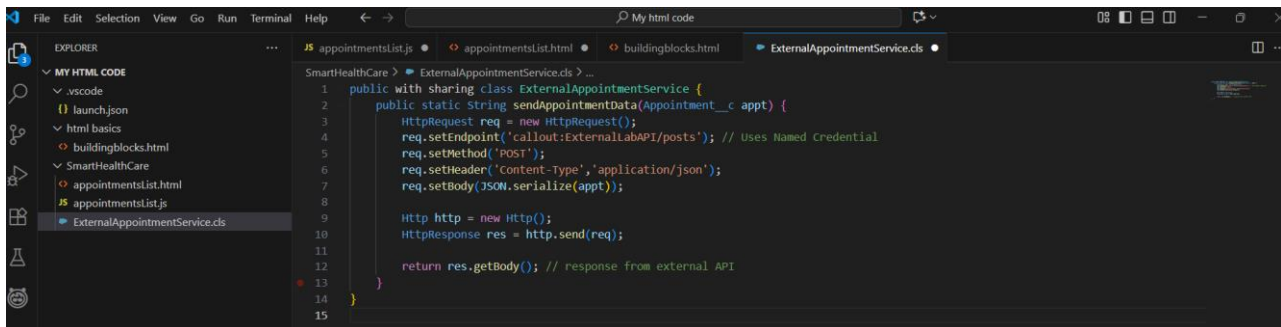
At the bottom right of the dialog are 'Cancel' and 'Save' buttons.

The screenshot shows the 'ExternalLabAPI' Named Credential detail page in the Salesforce Setup interface. The page is titled 'SETUP > NAMED CREDENTIALS' and 'ExternalLabAPI'. It contains the following fields and options:

- Label:** ExternalLabAPI
- Name:** ExternalLabAPI
- URL:** https://jsonplaceholder.typicode.com
- Enabled for Callouts:** ☒
- Authentication:**
  - External Credential:** PublicAPICredential
  - Client Certificate:** Search Certificates...
- Callout Options:**
  - Generate Authorization Header:** ☒
  - Allow Formulas in HTTP Header:** ☐
  - Allow Formulas in HTTP Body:** ☐

At the top right of the page are 'Edit' and 'Delete' buttons.

### 3.1.2 Apex Callout Using Named Credential



## 3.2 Platform Events

When a **new appointment is created**, the system should notify **external hospital systems** (e.g., waiting room display, ERP). Instead of polling APIs, Salesforce can publish a **Platform Event** in real time.

### 3.2.1 Create Platform Event

Platform Events enable **real-time communication** between Salesforce and external systems without constant polling. In this step, whenever a new appointment is created in the system, a **custom Platform Event** named `Appointment_Notification__e` is published. This event includes key appointment details like clinician name, patient name, and start time. External systems subscribed to this event—such as hospital displays, ERP systems, or patient notification services—

receive the information instantly. This approach improves operational efficiency, reduces latency, and ensures that critical updates are propagated immediately to all connected systems.

← → ↻ orgfarm-9c399a4996-dev-ed.develop.lightning.force.com/lightning/setup/EventObjects/page?address=%2F011%2Fe%3Fsetupid%3DEventObjects%26retURL%3D%252Fp%252Fs... ☆ Verify it's you

Setup Home Object Manager

plaf

- MuleSoft
  - Anypoint Platform Setup
- Einstein
  - Einstein Platform
  - Einstein Bots
  - Einstein.ai
- Custom Code
  - Platform Cache
- Integrations
  - Platform Events
- Security
  - Platform Encryption
    - Encryption Settings
    - Key Management

Didn't find what you're looking for? Try using Global Search.

### Platform Events

#### Platform Event Definition Edit

Save Save & New Cancel

**Platform Event Information**

Label Appointment Notification

Plural Label Appointment Notifications

Starts with vowel sound ☐

The object name is used when referencing the event via the API.

Object Name Appointment\_Notification\_e

Description

Event Type High Volume

Publish Behavior Publish Immediately

**Deployment Status** What is this?

☐ In Development

☒ Deployed

Save Save & New Cancel

### 3.2.2 Platform Event from Trigger

← → ↻ orgfarm-9c399a4996-dev-ed.develop.lightning.force.com/lightning/setup/EventObjects/page?address=%2F011gl000002OuPh%3FdetailType%3DFieldsAndRelationships%26setu... ☆ Verify it's you

Setup Home Object Manager

plaf

- MuleSoft
  - Anypoint Platform Setup
- Einstein
  - Einstein Platform
  - Einstein Bots
  - Einstein.ai
- Custom Code
  - Platform Cache
- Integrations
  - Platform Events
- Security
  - Platform Encryption
    - Encryption Settings
    - Key Management

Didn't find what you're looking for? Try using Global Search.

### Platform Events

#### Appointment Notification

Standard Fields (6) Custom Fields & Relationships (2)

**Platform Event Definition Detail** Edit Delete

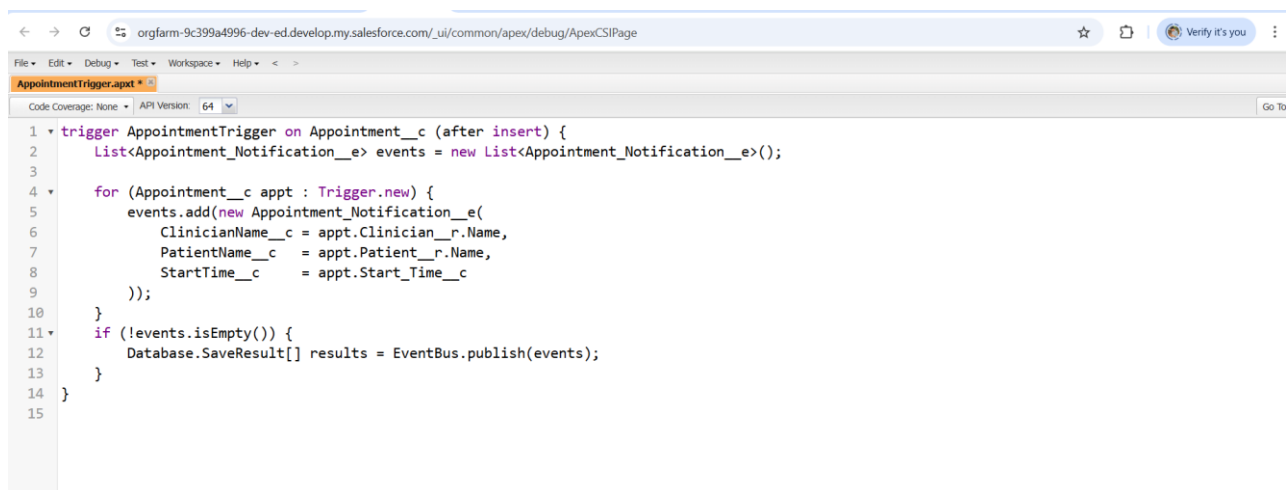
Singular Label	Appointment Notification	Description	
Plural Label	Appointment Notifications	Deployment Status	Deployed
Object Name	Appointment_Notification_e		
API Name	Appointment_Notification_e__e		
Event Type	High Volume		
Publish Behavior	Publish Immediately		
Created By	Palla Bhugabha, 9/27/2025, 9:14 AM	Modified By	Palla Bhugabha, 9/27/2025, 9:14 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	ClinicianName	ClinicianName__c	Text(50)			Palla Bhugabha, 9/27/2025, 9:15 AM

A screenshot of the Salesforce IDE interface. The browser address bar shows the URL: orgfarm-9c399a4996-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage. The IDE window title is 'AppointmentTrigger.apex'. The code editor shows the following Apex code:

```
1 trigger AppointmentTrigger on Appointment__c (after insert) {
2     List<Appointment_Notification__e> events = new List<Appointment_Notification__e>();
3
4     for (Appointment__c appt : Trigger.new) {
5         events.add(new Appointment_Notification__e(
6             ClinicianName__c = appt.Clinician__r.Name,
7             PatientName__c = appt.Patient__r.Name,
8             StartTime__c = appt.Start_Time__c
9         ));
10    }
11    if (!events.isEmpty()) {
12        Database.SaveResult[] results = EventBus.publish(events);
13    }
14 }
15
```

## 4. Expected Outcomes

---

- Named Credentials and External Credentials configured to securely manage external endpoints without exposing sensitive authentication details.
- Apex callouts successfully exchange appointment and compliance data with external healthcare systems.
- Platform Events established to provide real-time notifications of new appointments and compliance updates to external systems.
- Remote Site Settings and authentication measures implemented to ensure secure, authorized external access.
- Verified integration workflows demonstrating that the Smart Healthcare Hub can interoperate with external hospital services and APIs.

## 5. Conclusion

---

Phase 7 extended the Smart Healthcare Appointment & Compliance Hub by enabling secure integration with external systems. Through the use of Named Credentials, Apex callouts, and Platform Events, the system can now communicate with healthcare APIs and provide real-time updates to connected applications. By establishing secure access protocols and external connectivity, the hub is prepared to operate as part of a larger hospital ecosystem, ensuring interoperability, scalability, and improved healthcare service delivery.