

Phase 7 – Integration & External Access

Project: Event Management Portal

This phase focuses on enabling external integrations and API access for the Event Management Portal. Only the features implemented for this project are documented here.

1. External Credential

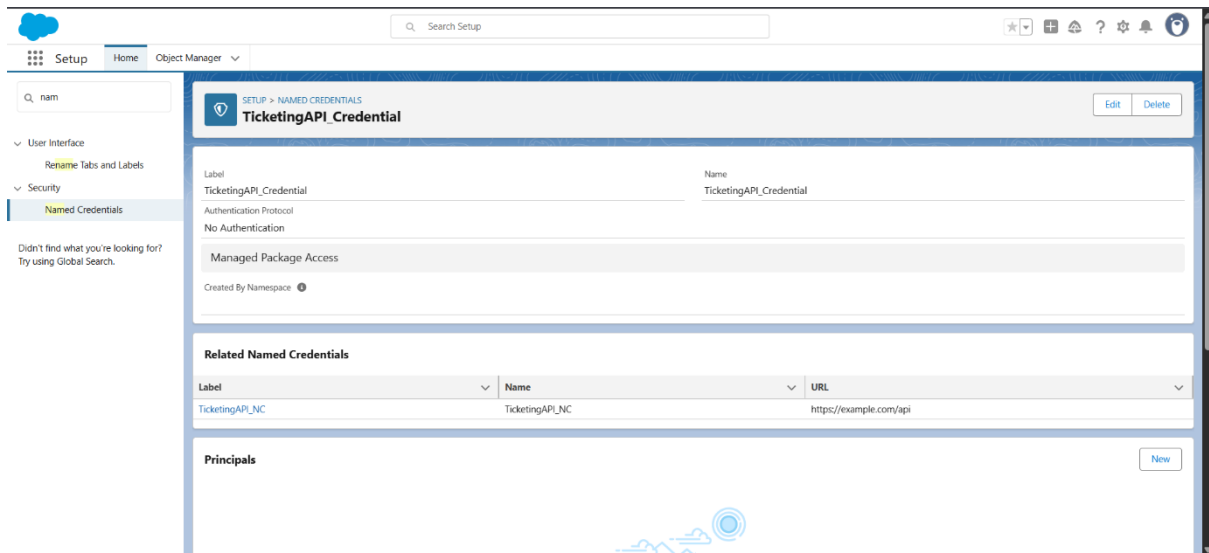
Purpose:

Defines authentication settings for external API callouts.

Steps Performed:

1. Navigated to **Setup** → **External Credentials** → **New**.
2. Entered the following:
 - **Label:** TicketingAPI_Credential
 - **Name:** TicketingAPI_Credential (auto-filled)
 - **Authentication Protocol:** No Authentication
3. Saved the credential.

Screenshot :



2. Named Credential

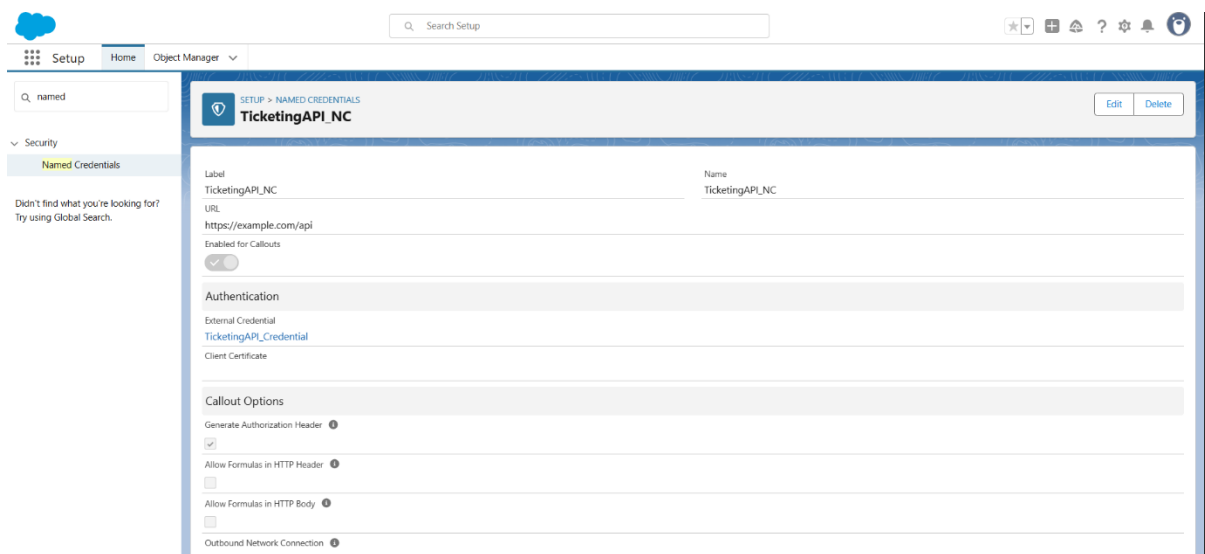
Purpose:

Provides the endpoint URL and links to the External Credential for making API callouts from Salesforce.

Steps Performed:

1. Navigated to **Setup → Named Credentials → New**.
2. Entered the following:
 - **Label:** TicketingAPI_NC
 - **Name:** TicketingAPI_NC
 - **URL:** <https://example.com/api>
 - **External Credential:** TicketingAPI_Credential
3. Saved the Named Credential.

Screenshot :



3. Principal Setup

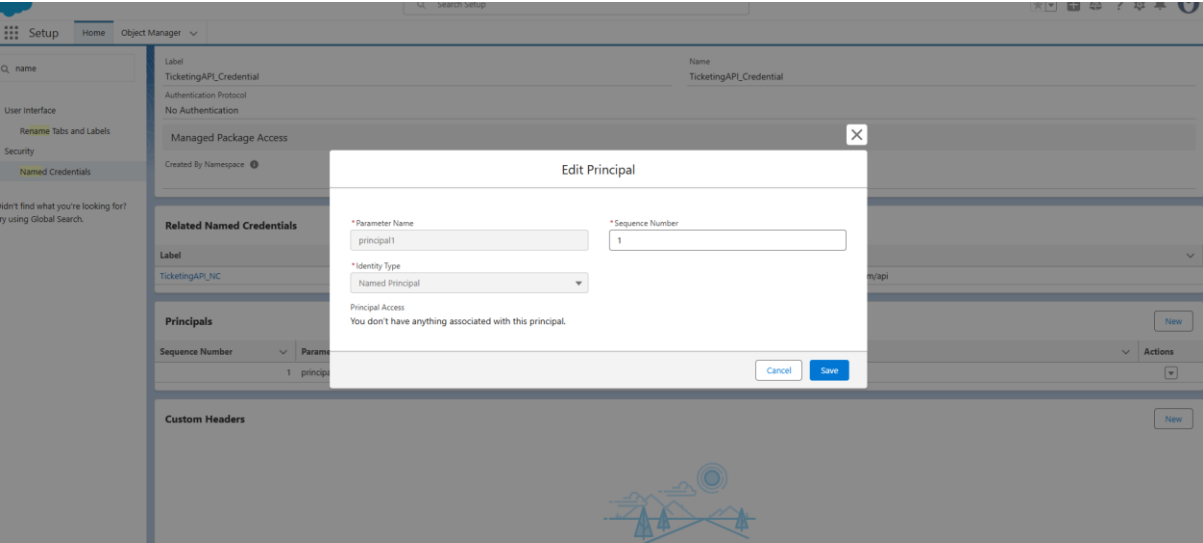
Purpose:

Configures access for the Named Credential using the External Credential.

Steps Performed:

1. In the External Credential setup, clicked **Create Principal**.
2. Entered the following:
 - **Parameter Name:** principal1
 - **Sequence Number:** 1
 - **Identity Type:** Named Principal
3. Saved the Principal configuration.

Screenshot :



4. Callout Test

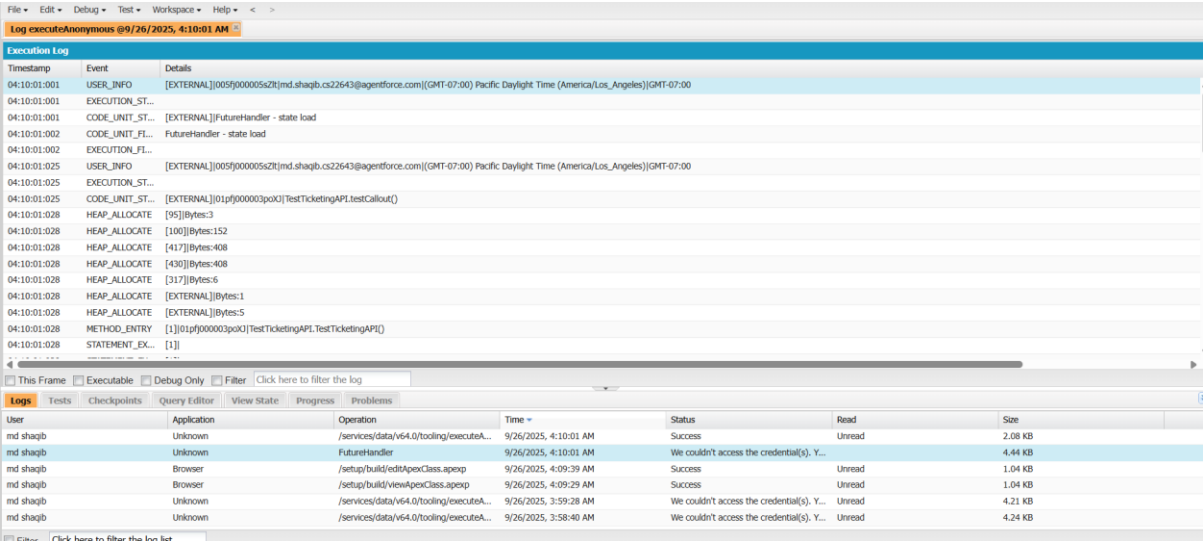
Purpose:

To ensure the Named Credential and External Credential work together and allow external API communication.

Steps Performed:

- 1. Created a test Apex class to call the external API using the Named Credential.
- 2. Verified the callout executed successfully in **Developer Console** → **Execute Anonymous** or **Debug Logs**.

Screenshot :



5. Platform Event

Purpose:

To notify external systems or other Salesforce processes when a Ticket record is updated.

Steps Performed:

1. Navigated to **Setup → Platform Events → New Platform Event**.
2. Created the event with the following properties:
 - **Label:** Ticket_Updates
 - **Plural Label:** Ticket_Updates
 - **Object Name:** Ticket_Updates
 - **Publish Behavior:** Publish Immediately
3. Added custom fields to track ticket details:
 - **Ticket ID:** Text(18)
 - **Status:** Text(50)
 - **Price:** Number(16,2)

Screenshot :

The screenshot displays the Salesforce Setup interface for configuring a Platform Event. The left sidebar shows the navigation menu with 'Platform Events' selected under 'Integrations'. The main content area is titled 'Platform Events' and shows the configuration for a new event named 'Ticket_Update'.

Platform Event Definition Detail:

Property	Value
Singular Label	Ticket_Update
Plural Label	Ticket_Updates
Object Name	Ticket_Update
API Name	Ticket_Update__e
Event Type	High Volume
Publish Behavior	Publish Immediately
Created By	md.shahib
Created Date	9/25/2025, 3:48 PM
Modified By	md.shahib
Modified Date	9/25/2025, 3:48 PM

Standard Fields:

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
Edit	Created By	CreatedBy	Lookup/User		
Edit	Created Date	CreatedDate	Date/Time		
Edit	Event ID	EventId	Text(18)		
Edit	Replay ID	ReplayId	External Lookup		

Custom Fields & Relationships:

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By	Modified Date
Edit	Price	Price__c	Number(16, 2)			md.shahib	9/25/2025, 3:51 PM
Edit	Status	Status__c	Text(50)			md.shahib	9/25/2025, 3:50 PM
Edit	Ticket ID	TicketID__c	Text(18)			md.shahib	9/25/2025, 3:50 PM

Triggers:

No triggers defined

6. Apex Test – Publish Platform Event

Purpose:

Tested the publishing of Platform Events using Apex to ensure proper event delivery.

Steps Performed:

1. Created a test Apex class TestTicketEvent with a method publishTicketUpdate() to publish sample events.

2. Executed the method and verified events were published successfully in **Developer Console** → **Logs**.

Screenshot :

The screenshot displays the Salesforce Setup interface. On the left, there is a navigation menu with 'Setup', 'Home', and 'Object Manager'. A search bar for 'apex class' is visible. The main content area shows the 'Apex Class Detail' for 'TestTicketingAPI'. The class is active, created by 'msd.shahab' on '9/25/2025, 3:27 PM', and has a code coverage of '0% (0/6)'. The 'Class Body' tab is selected, showing the following code:

```
1 public with sharing class TestTicketingAPI {
2     @future(callout=true)
3     public static void testCallout() {
4         Http http = new Http();
5         HttpRequest req = new HttpRequest();
6
7         // Use the Named Credential URL
8         req.setEndpoint('callout/TestTicketingAPI_NC'); // <-- Named Credential name
9         req.setMethod('GET'); // or POST if required
10
11         HttpResponse res = http.send(req);
12         System.debug('Response Status: ' + res.getStatusCode());
13         System.debug('Response Body: ' + res.getBody());
14     }
15 }
```

At the bottom of the class body, there are buttons for 'Edit', 'Delete', 'Download', 'Security', and 'Show Dependencies'.