📘 Phase 7: Integration & External Access

🎯 Objective

The goal of this phase is to integrate Salesforce with external systems, enable secure data communication using APIs, and implement event-driven mechanisms for seamless synchronization of customer success data.
This ensures that the CRM interacts smoothly with external services, monitors data changes, and maintains secure and authenticated access.

🔹 Modules Covered

- Named Credentials
- External Services
- Web Services (REST/SOAP)
- Callouts
- Platform Events
- Change Data Capture (CDC)
- Salesforce Connect
- API Limits
- OAuth & Authentication
- Remote Site Settings

🧩 Step 1: Remote Site Settings

📍 Purpose:

Remote Site Settings allow Salesforce to call external APIs securely. Without this setup, outbound callouts are blocked.

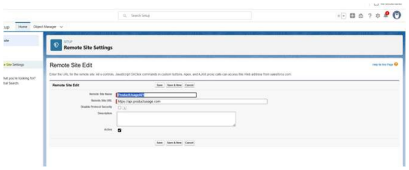
📜 Steps:

1. Navigate to **Setup → Security → Remote Site Settings → New Remote Site**
2. Enter details:
    - **Label:** ProductUsageService

- **Name:** ProductUsageService
- **URL:** https://api.productusage.com
- **Enabled for Callouts:** ✅

3. Click **Save**

🖼️ ***Screenshot Placeholder:***



🧩 Step 2: Named Credentials

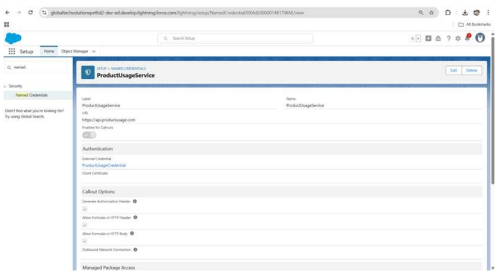
📍 Purpose:

Named Credentials simplify external API authentication by storing the endpoint URL and credentials securely.

📜 Steps:

1. Go to **Setup → Named Credentials → New Named Credential**
2. Enter details:
   - **Label:** ProductUsageService
   - **Name:** ProductUsageService
   - **URL:** https://api.productusage.com
   - **Enabled for Callouts:** ✅
   - **Authentication:** Select “None” *(or use an External Credential if integrated)*
3. Click **Save**

🖼️ ***Screenshot Placeholder:***



🧩 Step 3: External Services

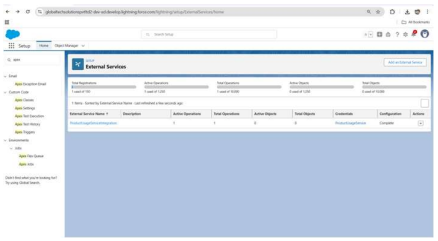
📍 Purpose:

External Services allow Salesforce to connect to third-party APIs using declarative tools, automatically generating Apex actions.

📜 Steps:

1. Go to **Setup → External Services → New External Service**
2. Select **Named Credential: ProductUsageService**
3. If available, upload the **OpenAPI/Swagger file** for your API.
4. Salesforce automatically generates **Apex Actions** for integration.

🖼️ *Screenshot Placeholder:*



🧩 Step 4: Web Services (REST API Callout)

📍 Purpose:

To demonstrate integration with an external REST API for product usage data.

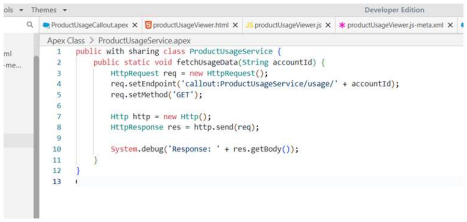
📜 Apex Example:

```
public with sharing class ProductUsageService {
    public static void fetchUsageData(String accountId) {
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:ProductUsageService/usage/' + accountId);
        req.setMethod('GET');
        Http http = new Http();
        HttpResponse res = http.send(req);
```

```
        System.debug('Response: ' + res.getBody());
    }
}
```

This Apex class fetches external data for an Account using the Named Credential setup.

🖼️ ***Screenshot Placeholder:***



🧩 Step 5: Platform Events

📍 Purpose:

Platform Events enable real-time communication inside Salesforce or with external systems. Here, they're used to notify other systems whenever an Account's Health Score changes.

📜 Steps:

1. Go to **Setup → Platform Events → New Platform Event**
2. Enter:
    - **Label:** Account_Health_Update
    - **API Name:** Account_Health_Update__e
3. Add fields:
    - AccountId__c → Text (18)
    - NewHealthScore__c → Number (3,2)
4. Save

📜 Apex Publisher Class:

```
public class HealthScoreEventPublisher {
    public static void publishEvent(List<Account> updatedAccounts) {
        List<Account_Health_Update__e> events = new List<Account_Health_Update__e>();
```

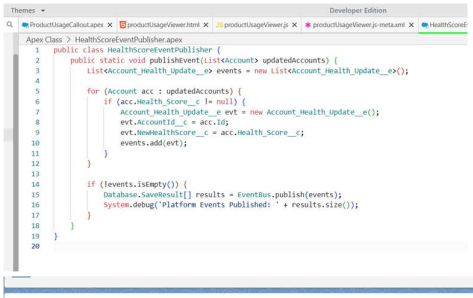
```
        for (Account acc : updatedAccounts) {
            if (acc.Health_Score__c != null) {
                Account_Health_Update__e evt = new Account_Health_Update__e();
                evt.AccountId__c = acc.Id;
                evt.NewHealthScore__c = acc.Health_Score__c;
                events.add(evt);
            }
        }

        if (!events.isEmpty()) {
            Database.SaveResult[] results = EventBus.publish(events);
            System.debug('Platform Events Published: ' + results.size());
        }
    }
}
```

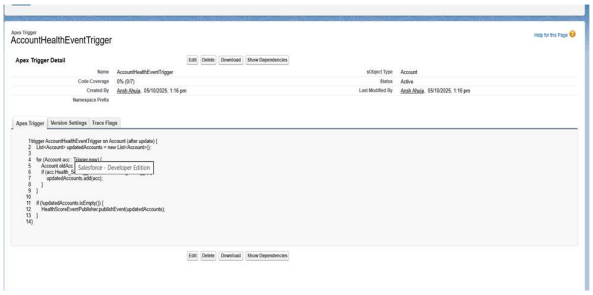
Trigger:

```
trigger AccountHealthEventTrigger on Account (after update) {
    List<Account> updatedAccounts = new List<Account>();

    for (Account acc : Trigger.new) {
        Account oldAcc = Trigger.oldMap.get(acc.Id);
        if (acc.Health_Score__c != oldAcc.Health_Score__c) {
            updatedAccounts.add(acc);
        }
    }

    if (!updatedAccounts.isEmpty()) {
        HealthScoreEventPublisher.publishEvent(updatedAccounts);
```

```
    }
}
```

🖼️ ***Screenshot Placeholder:***





🧩 Step 6: Change Data Capture (CDC)

📍 Purpose:

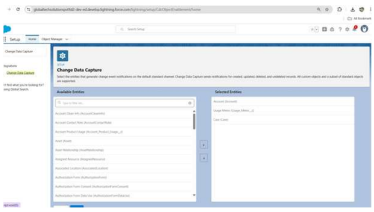
CDC allows automatic event publishing whenever a record changes, helping external systems stay updated.

🗒️ Steps:

1. Go to **Setup → Change Data Capture**
2. Click **Edit**
3. Select Objects:
    - Account
    - Usage_Metric__c
    - Support_Case__c
4. Click **Save**

Now, any change in these objects will trigger a CDC event.

🖼️ ***Screenshot Placeholder:***



🧩 Step 7: Salesforce Connect (Optional Integration)

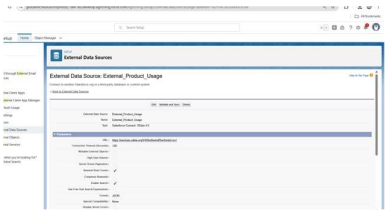
📍 Purpose:

Salesforce Connect allows access to external data in real time without storing it in Salesforce.

📜 Steps:

1. Go to **Setup → External Data Source → New**
2. Label: External_Product_Usage
3. Type: OData 4.0
4. URL: https://services.odata.org/V4/Northwind/Northwind.svc/
5. Save → Validate and Sync → Select tables → Sync

🖼️ *Screenshot Placeholder:*



🧩 Step 8: API Limits

📍 Purpose:

Salesforce enforces API request limits per 24 hours to ensure performance and security.

📜 Steps:

1. Go to **Setup → System Overview**
2. Scroll to **API Usage** section to monitor daily API consumption.

🧩 Step 9: OAuth & Authentication

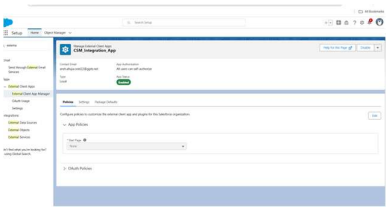
📍 Purpose:

OAuth ensures secure authorization for external applications connecting to Salesforce APIs.

📜 Steps:

1. Go to **Setup → App Manager → New Connected App**
2. Enter:
    - **App Name:** CSM_Integration_App
    - **API Name:** CSM_Integration_App
    - Enable OAuth Settings ✅
    - Callback URL: https://login.salesforce.com/services/oauth2/callback
    - Selected OAuth Scopes:
        - Full access (full)
        - Refresh token (offline access)
3. Click **Save**
4. Copy the **Consumer Key** and **Consumer Secret** for API authentication.

🖼️ *Screenshot Placeholder:*



🧩 Step 10: Verification and Testing

- ✅ Remote Site Settings and Named Credential created successfully
- ✅ REST API Callout executed successfully
- ✅ Platform Event fired when Account Health Score updated
- ✅ CDC enabled for Account, Usage Metric, and Support Case
- ✅ OAuth App created for secure access

- ✅ External data connection tested

📜 Phase 7 Summary

Feature	Description	Status
Remote Site Settings	Enables secure API callouts	✅ Completed
Named Credentials	Stores endpoint and credentials securely	✅ Completed
External Services	Connects to external APIs declaratively	✅ Completed
Web Services	REST callouts using Apex	✅ Completed
Platform Events	Event-driven updates for Account health	✅ Completed
Change Data Capture	Tracks object data changes	✅ Completed
Salesforce Connect	Links external data without import	✅ Completed
API Limits	Monitored via System Overview	✅ Completed
OAuth & Authentication	Enables secure access for external apps	✅ Completed

---

🧠 Conclusion

This phase established secure, real-time, and event-driven integration between Salesforce and external systems.
By implementing Named Credentials, Platform Events, and OAuth, the **Customer Success & Support Management CRM** achieved seamless data communication, automated synchronization, and enhanced interoperability — strengthening customer engagement and operational efficiency.
