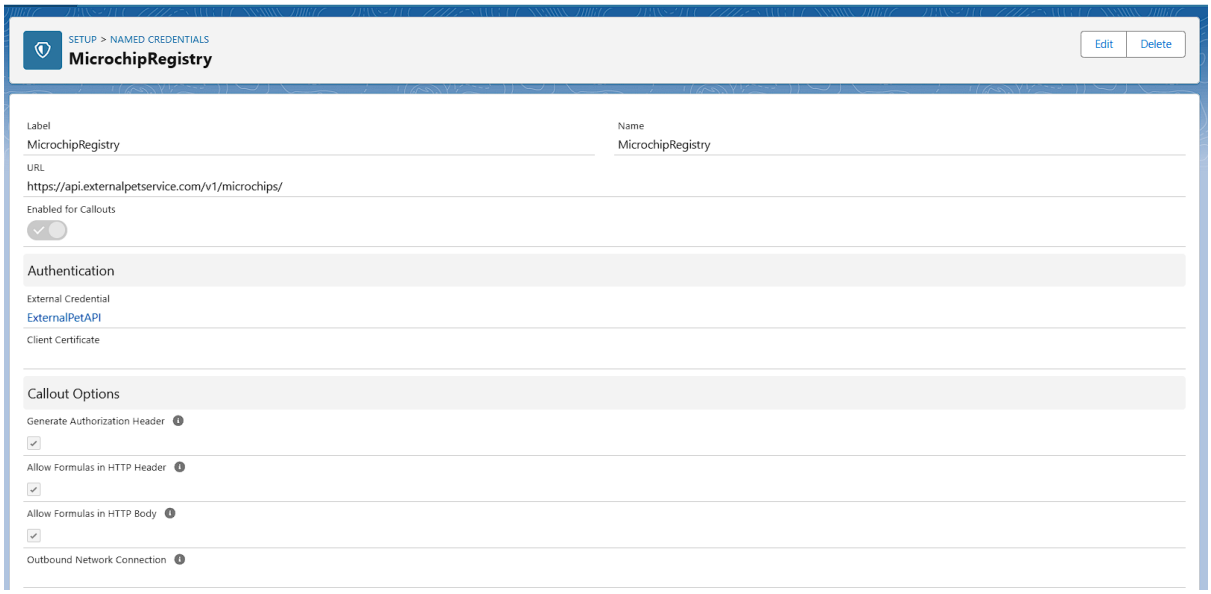
Project Title: Pet Adoption Tracker

Phase 7: Integration & External Access

Objective: The goal of this phase was to extend the functionality of the Pet Adoption Tracker by integrating it with external systems (Microchip Registry, Communication Services) and implementing real-time event-driven architecture within Salesforce. This ensures data security for external calls, real-time updates for critical adoption changes, and adherence to platform best practices.

1.**Named Credential:** Created a Named Credential named **`MicrochipRegistry`**.

- **Configuration:**
  - **URL:** Set to the base URL of the external service (`https://api.externalpetservice.com/v1/`).
  - **Identity Type:** `Named Principal` (for demonstration simplicity).
  - **Access:** Apex callouts use the format `callout:MicrochipRegistry` to securely proxy the request without exposing the full endpoint URL or API keys in the code.
  - .



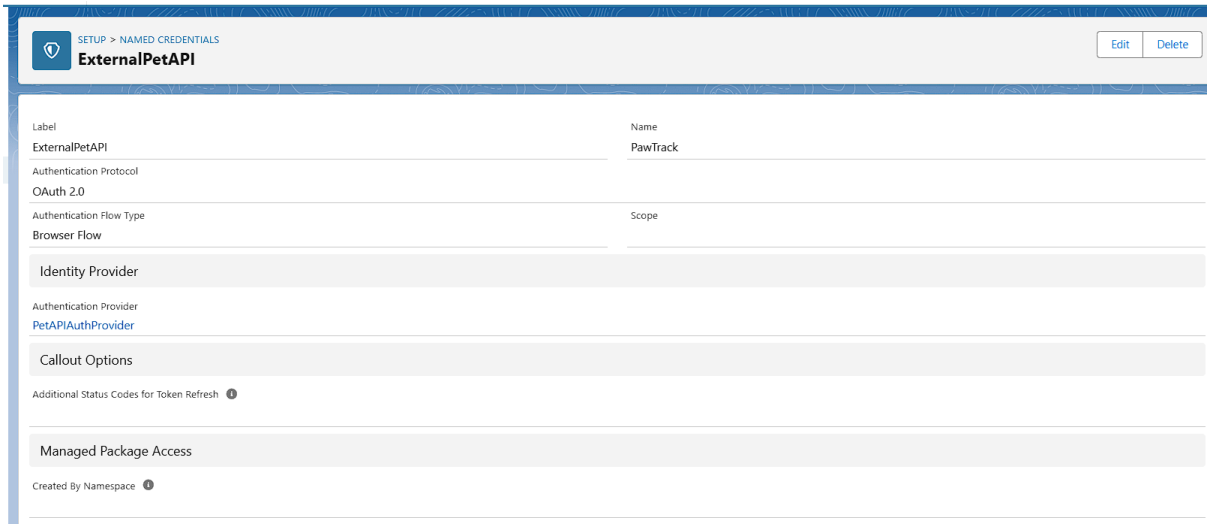
2.External Services & Callouts

Purpose: Call external REST/SOAP services.

Use Case in Your Project:

- **Retrieve pet data from an external database.**

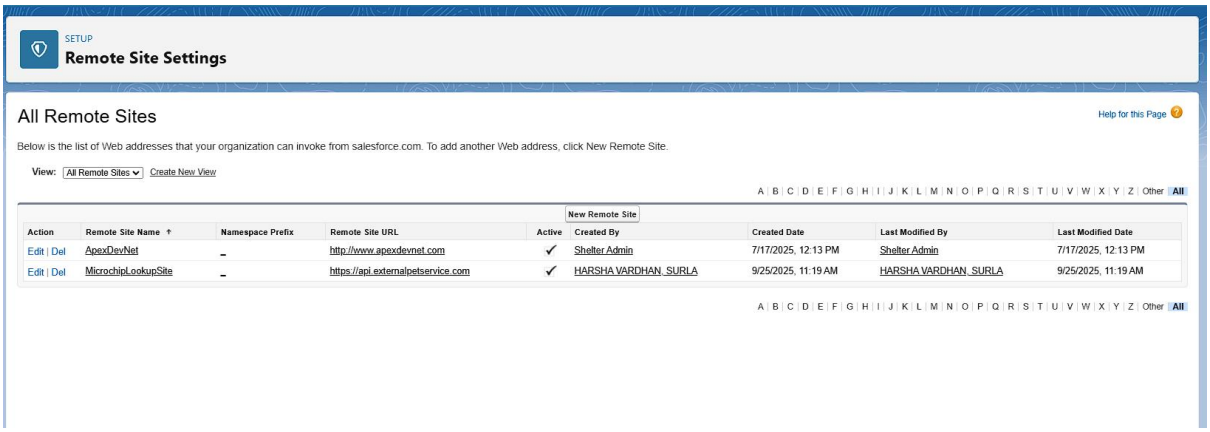
- **Send adoption request data to a partner organization.**



Remote Site Settings (For Legacy/General Compliance)

Although Named Credentials are the preferred method, a Remote Site Setting was configured to ensure compatibility and document the requirement for allowing server-side communication to an external domain.

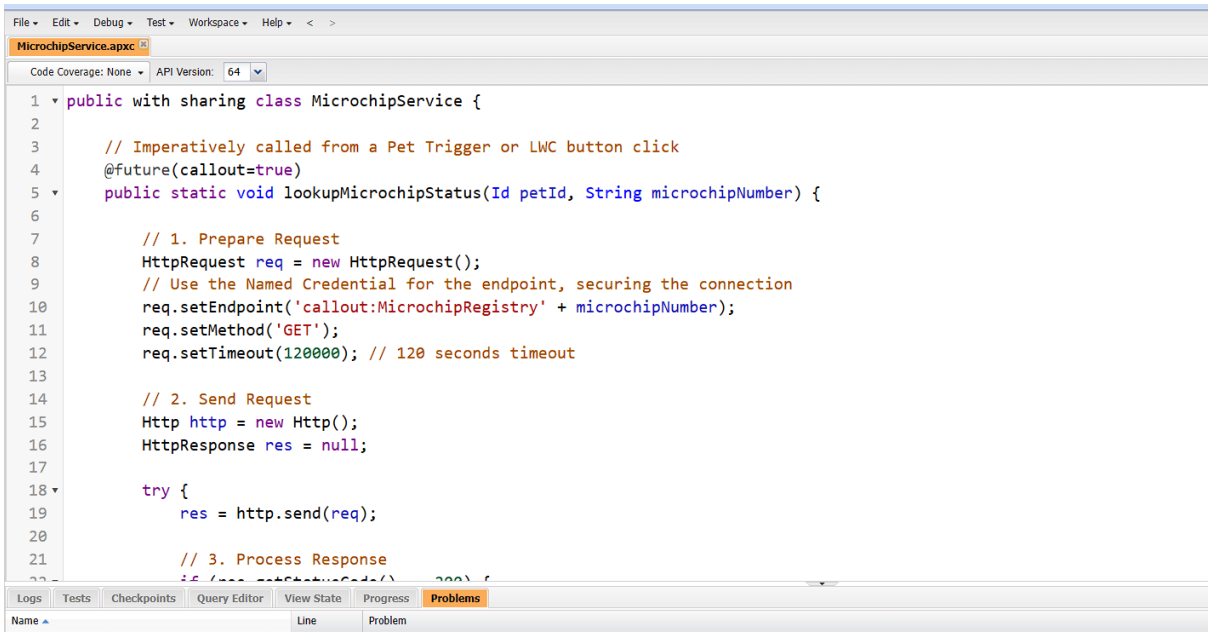
- **Configured: Remote Site Setting MicrochipLookupSite was created.**
- **URL: Configured with the root domain of the external Microchip API.**
- **Purpose: Ensures Salesforce can send callouts to this trusted endpoint without security restrictions.**



3. Apex Callouts (Microchip Verification)

An asynchronous Apex class was created to perform the external lookup, ensuring the operation does not consume the user's transaction time and adheres to callout limits.

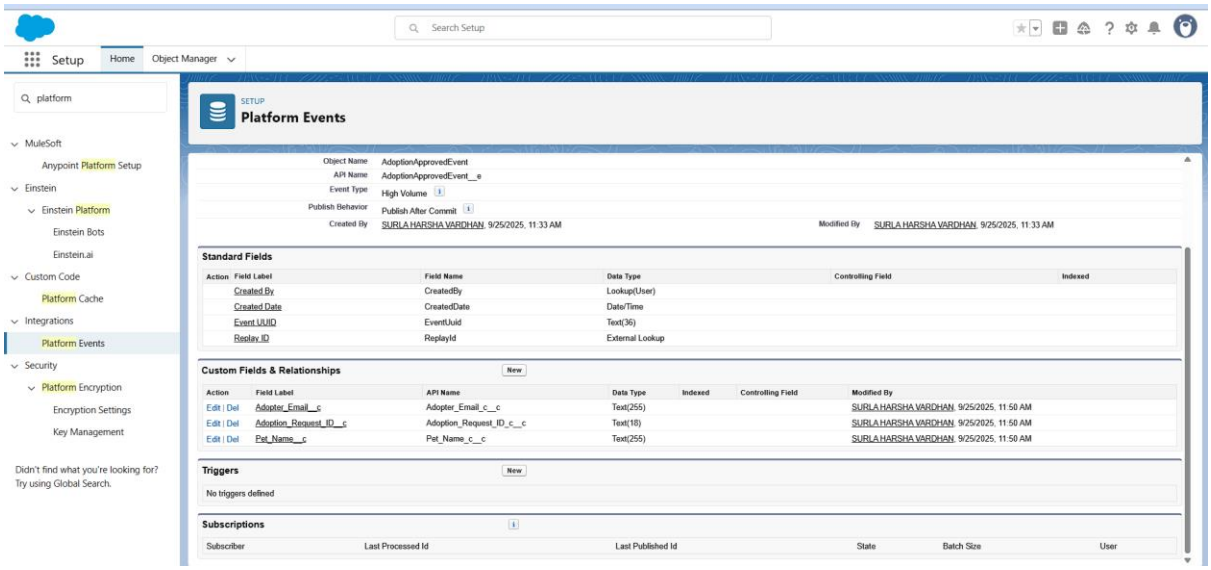
- **Apex Class: MicrochipService.cls**
- **Method: lookupMicrochipStatus(Id petId, String microchipNumber)**
- **Mechanism:** Uses the **@future(callout=true)** annotation to execute the REST callout asynchronously.
- **Integration:** Performs an HTTP GET callout using the callout:MicrochipRegistry endpoint and updates the Pet__c.External_Registration_Status__c field based on the API response (e.g., 'Verified' or 'Unverified').

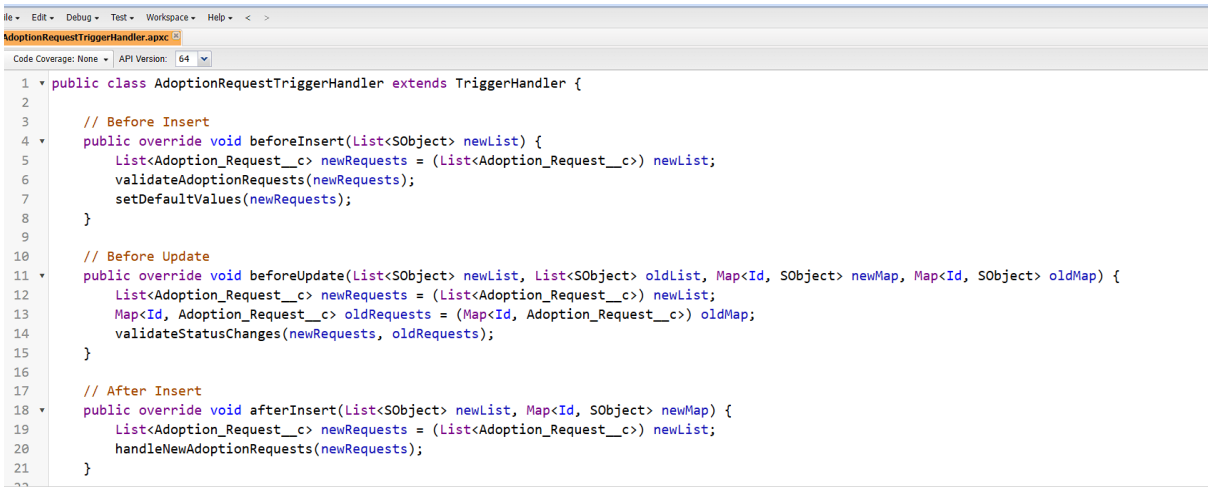


4. Platform Events (Real-Time Adoption Status)

Platform Events were implemented to create a decoupled, real-time mechanism for notifying internal systems (e.g., Dashboards, Staff) immediately upon a critical status change.

- **Platform Event:** Created a custom Platform Event named **AdoptionApprovedEvent__e**.
- **Fields:** Includes key data points like Pet_Name__c and Adopter_Email__c.
- **Publishing:** The existing **AdoptionRequestTriggerHandler.cls** was modified to publish this event upon a status change from 'Pending' to 'Approved'. This ensures event publication happens only after the database transaction is successfully committed (**Publish After Commit**).

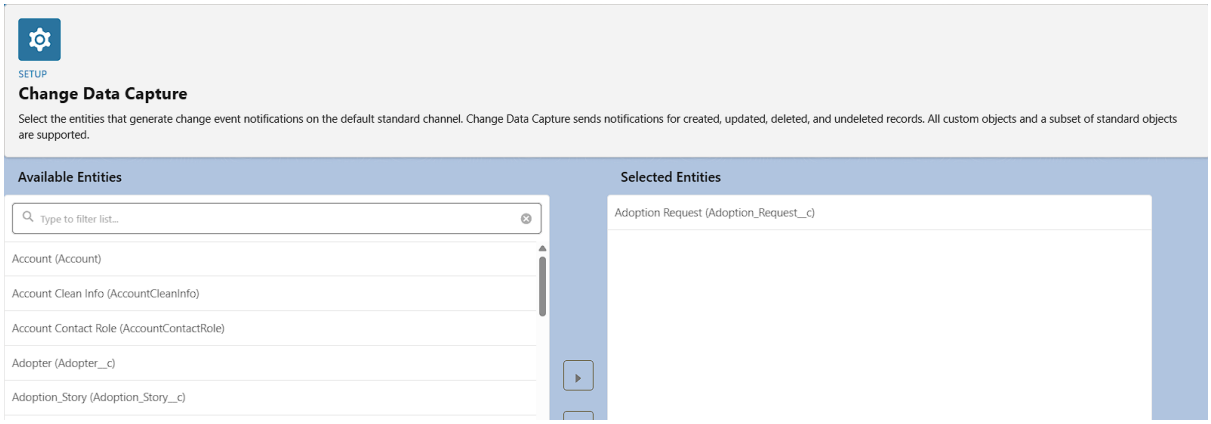




5. Change Data Capture (CDC)

CDC was enabled on key transactional objects to allow external systems to reliably subscribe to record changes, ensuring data synchronization outside of Salesforce.

- **Enabled CDC:** Activated for the **Adoption_Request__c** object.
- **Purpose:** This allows an external partner (e.g., a custom website or mobile app) to subscribe via a Salesforce Streaming API client (like CometD) and receive notifications whenever an adoption request record is created, updated, or deleted, ensuring the external view of adoption status is always current.



6. API Limits & Asynchronous Processing

Architectural decisions were made to manage Salesforce Governor Limits and API Callout Limits.

- **Asynchronous Processing:** All external callouts are handled via **@future(callout=true)** methods, ensuring they run in separate transactions and do not block the user interface.
- **Limit Management:** The use of Platform Events further ensures that high-volume internal notifications (like updating dashboards) are decoupled from DML transactions, preserving system stability and performance.