**Detailed Documentation for ConsentApplication Flow**

**Overview**

This document provides a step-by-step explanation of the ConsentApplication flow based on the provided sequence diagram. Each step includes details such as the type of call (GET, POST, Redirect), payload, query parameters, expected responses, and any pending decisions based on the meeting discussions.

**Steps in the Diagram**

**1. Lender User Login**

* **Action:** LenderUser logs in to the TSPApplication.
* **Call Type:** POST
* **Payload:**
* {
* "username": "lender\_username",
* "password": "lender\_password"
* }
* **Response:** TSPApplication authenticates the user and redirects to PingFederate for OIDC login.

**2. PingFederate OIDC Authentication**

* **Action:** PingFederate handles OIDC login.
* **Call Type:** Redirect
* **Query Parameters:** N/A (Handled internally within PingFederate)
* **Expected Behavior:** PingFederate validates user credentials.

**3. Check Lender-TSP Relationship**

* **Action:** PingFederate checks the relationship between the lender and the TSP.
* **Call Type:** GET
* **Endpoint:** PingFederate -> PostgresDatabase
* **Query Parameters:**
  + lender\_id: Identifier for the lender.
  + tsp\_id: Identifier for the TSP.
* **Response:**
  + If the relationship exists: Proceed to ConsentApplication with a reference (REF).
  + If the relationship does not exist: Display an access denied page.

**Relationship Exists**

**4. Attribute Pickup**

* **Action:** ConsentApplication retrieves user attributes from PingFederate using the REF.
* **Call Type:** GET
* **Endpoint:** PingFederate Agentless Attribute Pickup
* **Query Parameters:**
  + REF: Reference ID from PingFederate.
* **Response:**
* {
* "user\_id": "test09",
* "attributes": {
* "email": "user@example.com",
* "name": "John Doe"
* }
* }

**5. Get System IDs**

* **Action:** ConsentApplication queries the CIAM\_API for system IDs associated with the user.
* **Call Type:** GET
* **Endpoint:** CIAM\_API
* **Query Parameters:**
  + user\_id: Identifier for the user.
* **Response:** List of system IDs (not documented due to missing details).

**6. Read Consent**

* **Action:** ConsentApplication checks if consent already exists in the datastore.
* **Call Type:** GET
* **Endpoint:** /consent/v1/consents
* **Query Parameters:**
  + actor: User identifier (e.g., test09).
  + subject: System identifier (e.g., bbSystemId23).
  + definition: Consent definition ID (e.g., consent3).
* **Response:**
  + If consent exists:
  + {
  + "\_embedded": {
  + "consents": [
  + {
  + "id": "0a4d1471-1806-46a1-afa2-b8265c94d9e7",
  + "status": "accepted",
  + "subject": "bbSystemId23",
  + "actor": "test09",
  + "createdDate": "2025-01-28T15:48:31.619Z"
  + }
  + ]
  + }
  + }
  + If consent does not exist:
  + {
  + "count": 0,
  + "size": 0
  + }

**7. Attribute Dropoff (If Consent Exists)**

* **Action:** ConsentApplication performs an attribute dropoff with PingFederate.
* **Call Type:** POST
* **Endpoint:** PingFederate Agentless Attribute Dropoff
* **Payload:**
* {
* "user\_id": "test09",
* "status": "consent exists"
* }
* **Response:**
  + New REF is returned by PingFederate.
* {
* "REF": "new\_reference\_id"
* }
* **Redirect:** The user is redirected back to the TSPApplication with the new REF.

**Consent Does Not Exist**

**8. Display UI Page**

* **Action:** ConsentApplication displays the consent UI to the user.
* **Behavior:** User views the consent text, selects system IDs, and submits a response (agree/deny).

**9. Save Consent (User Agrees)**

* **Action:** ConsentApplication creates a new consent entry in the datastore.
* **Call Type:** POST
* **Endpoint:** /consent/v1/consents
* **Payload:**
* {
* "status": "accepted",
* "subject": "bbSystemId23",
* "actor": "test09",
* "audience": "FMSSO-FANNIEMAE",
* "definition": {
* "id": "consent3",
* "version": "1.4",
* "locale": "en-US"
* },
* "dataText": "consent3",
* "purposeText": "consent3"
* }
* **Response:**
* {
* "id": "ffa1290d-6f30-4777-a464-fe3f1468351c",
* "status": "accepted",
* "createdDate": "2025-02-07T19:48:31.594Z"
* }
* **Follow-Up:** Perform Attribute Dropoff and redirect to TSPApplication.

**10. User Denies Consent**

* **Action:** Display deny message page and redirect back to PingFederate.
* **Call Type:** POST (for Attribute Dropoff)
* **Payload:**
* {
* "user\_id": "test09",
* "status": "denied"
* }
* **Response:** New REF returned by PingFederate, user redirected back with denial status.

**Pending Decisions Based on Meeting Notes**

1. **Error Handling:**
   * Confirm where the error page should be displayed (PingFederate vs. ConsentApplication).
2. **Session Termination:**
   * Clarify whether session termination occurs at PingFederate or ConsentApplication level after denial.
3. **Dynamic Database Support:**
   * Evaluate transitioning from Postgres to DynamoDB for future scalability.
4. **Authentication Mechanism:**
   * Finalize whether to use basic auth, token-based auth, or other mechanisms for securing API calls.
5. **Error Messaging for Denial:**
   * Define the payload structure to communicate denial back to PingFederate accurately.