README — Verify Benefits (CBVR)

Deployment & Testing Step-by-step Guide

What this app does (quick)

- Users create a **CareBenefitVerifyRequest__c (CBVR)** record.
- A **Screen Flow** ("Verify Benefits (CBVR) – UI") calls **BV_VerifyService** to send a verification request (JSON) to the external endpoint (via **BV_Client**).
- The external API returns JSON. Flow calls **BV_ResponseParser_Invocable** to parse it.
- Flow then **updates the CBVR**:

  Status__c = Acknowledged, Status_Reason__c = <from API or default>, **clears** Last_Error__c, and optionally sets External_Request_Id__c.

Deploying — step-by-step

1) Create the custom object and fields

Object: CareBenefitVerifyRequest__c (Label "CareBenefitVerifyRequest")

Create the following (match API Names & types exactly):

- **ServiceType__c** (Picklist) — e.g., values: Medical, Dental, Infusion
- **Service_Date__c** (Date)
- **ICD10__c** (Text 10)
- **CPT__c** (Text 10)
- **Patient_Details__c** (Lookup → Account) **(Required)**
- **Patient_First_Name__c** (Text 80)
- **Patient_Last_Name__c** (Text 80)
- **Date_of_Birth__c** (Date)
- **Gender__c** (Picklist) — Male, Female, Other
- **Insurance_Information__c** (Lookup → MemberPlan) **(Required)**
- **Insurance_Provider_Name__c** (Text 120)
- **Policy_Number__c** (Text 30)
- **Group_Number__c** (Text 30)
- **Subscriber_Id__c** (Text 30)
- **Provider_Information__c** (Lookup → Account) **(Required)**
- **Provider_NPI__c** (Text 10)

- **Provider_First_Name__c** (Text 80)
- **Provider_Last_Name__c** (Text 80)
- **Status__c** (Picklist) — Acknowledged, Error, …
- **Status_Reason__c** (Long Text)
- **External_Request_Id__c** (Text 80)
- **Last_Error__c** (Long Text)

2) (Optional) Coverage result object

Object: CoverageBenefit__c

Fields commonly used:

- **CareBenefitVerifyRequest__c** (Lookup → CBVR)
- **External_Request_Id__c** (Text 80)
- **Status__c** (Picklist), **Status_Reason__c** (Long Text)

3) Page Layout & Record Page

- **CareBenefitVerifyRequest Layout**: include the fields above plus the **Verify Benefits** action.
- **CareBenefitVerifyRequest Record Page**: Lightning record page that surfaces the action in the Highlights panel.

In your org these already exist: **CareBenefitVerifyRequest Layout** and a **CareBenefitVerifyRequest Record Page**.

4) Queues (workload routing)

Create queues that support **CareBenefitVerifyRequest** (Setup → *Queues* → New):

- **Care Representatives**, **Dental**, **Emergency**, **Medical**

  Add CBVR to *Supported Objects* and add users to each queue.

  Use these queues for triage/ownership of requests; users can "Accept" from the queue list view.

5) Named Credential (callout)

Create a Named Credential pointing to your mock/real endpoint:

- **Label/Name:** BV_Mock (or your choice)
- **URL:** e.g. https://infinitusmockbvendpoint-rji9z4.5sc6y6-2.usa-e2.cloudhub.io/benefit-verification-request
- **Auth:** HTTP Basic (Username test_user, Password test_password)

Ensure **BV_Client.verify(String body)** uses callout:BV_Mock/benefit-verification-request.

6) Apex classes (deploy or copy)

- BV_VerifyService — builds JSON, validates inputs, calls BV_Client, maps response, writes Status/Reason/Last_Error to CBVR.
- BV_ResponseParser_Invocable — parses JSON (or falls back to CBVR fields).
- BV_Client — low-level HTTP callout using the Named Credential.
- BV_VerifyServiceTest — mocks + end-to-end test coverage.

7) Flow — Verify Benefits (CBVR) – UI

Flow steps and wiring:

1. **Apex Action: Verify Benefits** (API Name BV_VerifyService.run)
    - **Input:** recordId (the CBVR Id)
    - **Useful outputs:** responseBody, statusCode, error
2. **Apex Action: Parse Verify JSON** (BV_ResponseParser_Invocable.run)
    - **Inputs:**
        - cbvrId = {!recordId}
        - responseBody = {!Verify_Benefits.responseBody} *(optional; parser can fall back to CBVR)*
3. **Update Records (CareBenefitVerifyRequest)**
    - **Filter:** Record ID **Equals** {!recordId}
    - **Set:**
        - Status__c = "Acknowledged"
        - Status_Reason__c = {!Parse_Verify_JSON.statusReason} with a formula fallback if blank:

```
IF(
  ISBLANK({!Parse_Verify_JSON.statusReason}),
  "Care Benefit Verification Request successfully sent to Benefits Verification Provider.",
  {!Parse_Verify_JSON.statusReason}
)
```

        - Last_Error__c = GlobalConstant.Null
        - (Optional) External_Request_Id__c = {!Parse_Verify_JSON.externalRequestId}

8) Action button

Object Manager → **CareBenefitVerifyRequest** → *Buttons, Links, and Actions* →

Create / confirm **Action (Flow) "Verify Benefits"** pointing to the flow above.

Add it to the **CareBenefitVerifyRequest Layout** and confirm it appears on the **CareBenefitVerifyRequest Record Page**.

9) Permissions

Grant users:

- **Run Flows**, **Run Apex**, access to the **Named Credential**
- CRUD/FLS on the CBVR fields updated by the Flow
- Membership in relevant **Queues** (if used)

Testing — quick start

A) Execute Anonymous (Developer Console or Workbench → Apex Execute)

A1. Check the Id

```
// Replace with the Id in your CBVR URL (e.g., a2Va5...):
Id cbvrId = 'a2Va50000037KOTEA2';
System.debug( JSON.serializePretty( BV_VerifyService.verify(cbvrId) ) );
```

A2. Get the details

```
// Replace with the Id in your CBVR URL (e.g., a2Va5...):
Id cbvrId = 'a2Va50000037KOTEA2';
List<BV_VerifyService.ResultItem> out = BV_VerifyService.verify(cbvrId);
System.debug('Verify returned statusCode=' + out[0].statusCode + ', error=' + out[0].error);
```

A3. Sanity-parse a sample JSON like the parser does
(This doesn't invoke the invocable class — just shows the same approach.)

```
String sample = '{"status":"Acknowledged","statusReason":"Care Benefit Verification Request successfully sent to Benefits Verification Provider.","externalRequestId":"EV-TEST-1062129710","requestId":"a2vXXXXXXXXXXX","serviceType":"Medical","serviceDate":"2025-10-10"}';
Map<String,Object> root = (Map<String,Object>) JSON.deserializeUntyped(sample);
System.debug('status=' + (String)root.get('status'));
System.debug('statusReason=' + (String)root.get('statusReason'));
System.debug('externalRequestId=' + (String)root.get('externalRequestId'));
```

B) Workbench — create a CBVR via REST (POST)

Endpoint:

/services/data/v61.0/sobjects/CareBenefitVerifyRequest__c

Headers: Content-Type: application/json

B) Workbench REST Explorer

1) Grab the the Ids in Workbench

SELECT Id, Name FROM Account WHERE Name LIKE 'Test Patient%' LIMIT 1

SELECT Id, Name FROM MemberPlan WHERE Name = 'BVR-0001' LIMIT 1

(If you prefer REST Explorer for this step, use GET on

/services/data/v61.0/query/?q=SELECT+Id,Name+FROM+Account+WHERE+Name+LIKE+'Test+Patient%25'+LIMIT+1

and similar for the others.)

*(If you keep the method public, you can still use Workbench to create the CBVR, then click the **Verify Benefits** button in the UI to run the screen flow.)*

---

2) Sample external response (for slides or narration)

SELECT Id FROM CoverageBenefit__c WHERE External_Request_Id__c = 'EV-MANUAL-1'

Say (clarify):

"In this org the screen flow is designed for UI runs, so from Workbench I usually just create the CBVR by REST and then click the button in the UI to run the verification."

Sample request/response (what the Apex sends/expects)

Request body sent by BV_VerifyService:


```
{
 "requestId": "a2Va5000003670JEAQ",
 "externalRequestId": "BVR-a2Va50000038qSjEAI",
 "name": "BVR-0001",
 "serviceType": "Medical",
 "serviceDate": "2025-10-10",
 "icd10": "Z00.00",
 "cpt": "99213",
 "patient": { "firstName": "John", "lastName": "Doe", "dateOfBirth": "01-01-1995", "gender": "Male" },
 "insurance": { "providerName": "Aetna", "policyNumber": "P-123", "groupNumber": "G-123", "subscriberId":
"SUB-0001" },
 "provider": { "npi": "1234567890", "firstName": "Alice", "lastName": "Smith" }
}
```


Success response (expected)


```
{
 "status": "Acknowledged",
 "statusReason": "Care Benefit Verification Request successfully sent to Benefits Verification Provider.",
 "externalRequestId": "EV-TEST-1062129710"
}
```


What you should see on the CBVR after running the Flow

- Status__c = Acknowledged
- Status_Reason__c = “Care Benefit Verification Request successfully sent to Benefits Verification Provider.” *(or API text)*
- Last_Error__c = (blank)
- External_Request_Id__c = EV-TEST-1062129710 *(if mapped)*

Error variants

- **Validation error (from our guardrails):**
  - Status__c = Error, Status_Reason__c = Validation failed, Last_Error__c lists the missing/invalid fields.
- **HTTP 500 from vendor:**
  - Status__c = Error, Status_Reason__c = Internal Server Error.

Where things live (quick map)

- **Action: Verify Benefits** (Object: *CareBenefitVerifyRequest*) → launches the Flow.
- **Layout: CareBenefitVerifyRequest Layout** → includes the action & key fields.
- **Record Page: CareBenefitVerifyRequest Record Page** → Lightning page used for the object.
- **Queues: Care Representatives / Dental / Emergency / Medical** → each supports *CareBenefitVerifyRequest*. Use for routing & ownership before/after verification.

Tips / Troubleshooting

- **Flow fault**: usually Named Credential path/permissions or missing recordId in debug runs.
- **"Verify_Benefits resource doesn't exist"**: reselect the flow action output variable after you rename elements.
- **REST insert fails with REQUIRED_FIELD_MISSING**: provide the three required lookups (Patient_Details__c, Insurance_Information__c, Provider_Information__c).
- **No status update**: in the Update Records step, filter **Record ID Equals {!recordId}** (not "All records").