README — Verify Benefits

Deployment & Testing Step-by-step Guide

1) Prerequisites

- Salesforce org with Health Cloud.
- Salesforce CLI installed (sf --version).
- Admin permissions in the target org.

2) Metadata & Configuration Overview (what this package includes/expects)

- Standard & Custom Objects / Fields used by the solution
  - **Account (Patient/Provider)** with fields like FirstName, LastName, PersonBirthdate, Gender__c, NPI__c.
  - **MemberPlan** with Insurance_Provider_Name__c, Policy_Number__c, Group_Number__c, Subscriber_ID__c.
  - **CareBenefitVerifyRequest (CBVR)** with External_Status__c, External_Status_Reason__c, Request_Payload__c, Response_Payload__c, Service_Type__c, Service_Date__c, Diagnosis_Code__c, Procedure_Code__c, Patient__c (Lookup with Accounts), Provider__c (Lookup with Accounts) and MemberPlanId (lookups with MemberPlan).
  - **CoverageBenefit (created after verification**) with CareBenefitVerifyRequestId (Lookup with CareBenefitVerifyRequest), Verification_Status__c, Status_Reason__c, API_Response_Ref__c.
  - **Integration_Log__c** for end-to-end request/response and error logging (fields include Name, Request_Type__c, Related_Request__c (Lookup with CareBenefitVerifyRequest), Request_JSON__c , Response_JSON__c, Status_Code__c, Error_Message__c, Attempted_On__c).
  - **CBVR_Work_Item__c** work wrapper for queue routing (CareBenefitVerifyRequest__c (Lookup with CareBenefitVerifyRequest), Status__c, Benefit_Type__c, OwnerId (Lookup with User/Queue Priority__c, CreatedDate).
- Security & Integrations
  - **Permission Set:** Benefit_Verification_Permissions (assign to users).
  - **External Credential:** Benefit_Verification_External_Credential.
  - **Named Credential:** Benefit_Verification_NC.
- Runtime Components
  - **Apex Trigger:** CareBenefitRequestTrigger on CareBenefitVerifyRequest.
  - **Apex Classes**: CareBenefitService, CareBenefitAPIHandler, IntegrationLogger, CareBenefitResultAPI.
  - **Queues:** Consultation Verification Queue, Pharmacy Verification Queue, Medical Verification Queue.

---

3) Deployment Steps (SFDX)

1. **Authorize the target org**
   sf org login web -a hc-target

2. **Deploy metadata (adjust path if your repo structure differs)**
   sf project deploy start -o hc-target -d force-app --ignore-conflicts

3. **Create/verify external identity**
   Create External Credential Benefit_Verification_External_Credential and Named Credential Benefit_Verification_NC and wire them to your API provider per your environment secrets/policies. (Names must match exactly.)

4. **Assign permissions**
   # In Setup: Assign Permission Set "Benefit_Verification_Permissions" to intended users (Or via User Assignments in Setup.)
5. **Create queues (if not already present) and set routing rules as needed:**
   - Consultation Verification Queue
   - Pharmacy Verification Queue
   - Medical Verification Queue

---

4) Seed/Test Data (Patients, Providers, Plans)

Create minimally required reference data:

- Patient (Account) with demographic fields (DOB, Gender, etc.).
- Provider (Account) with NPI__c.
- MemberPlan with Insurance_Provider_Name__c, Policy_Number__c, etc.

Quick create via Apex (Dev Console > Execute Anonymous):

```
(
CareBenefitService.createVerificationRequest(
   [SELECT Id FROM Account WHERE Name = 'Test2 Patient2' LIMIT 1].Id,
   [SELECT Id FROM Account WHERE Name = 'Test2 Provider2' LIMIT 1].Id,
   [SELECT Id FROM MemberPlan WHERE Name = 'MP2' LIMIT 1].Id,
   'Medical',
   Date.today(),
   'A09',
   '99213'
);
```

This constructs a CareBenefitVerifyRequest with the service context (type/date, ICD-10 and CPT) and links to Patient/Provider/Plan.

---

5) How Verification Flows End-to-End

1. Create CBVR (manually, flow, or via CareBenefitService.createVerificationRequest).
    - The system stamps Request_Payload__c (JSON sent out) and later Response_Payload__c.
    - Integration_Log__c records outbound/inbound attempts, status codes, and any error messages.
2. Outbound call is performed using Benefit_Verification_NC; the IntegrationLogger captures request/response.
3. Inbound result (webhook / callback) updates the originating CBVR; when completed, a CoverageBenefit record is created and stamped with verification status & trace reference.
4. Work routing (optional) via CBVR_Work_Item__c to the appropriate queue based on benefit type.

---

6) How to Test (Manual + API)

A) Smoke Test (UI or Apex)

- Run the Apex snippet above to create a CBVR. Confirm:

    - CBVR shows your service details (Service_Type__c, Service_Date__c, Diagnosis_Code__c, Procedure_Code__c) and lookups to Patient/Provider/MemberPlan.
    - An Integration Log entry gets created on send/receive attempts.

B) Simulate Inbound Callback (Workbench → REST Explorer)

Send a POST to your exposed REST resource (e.g., the class handling inbound result). Use the sample payload from the spec:

```
{
 "CareBenefitVerifyRequestId": "0n2fn0000000HvJAAU",
 "Name": "testReq2",
 "Verification_Status__c": "Acknowledged",
 "Status_Reason__c": "All details verified",
 "MemberPlanId": "0Sqfn0000000HSHCA2",
 "API_Response_Reference__c": "test"
}
```

Expected behavior:

- The target CBVR is updated (status/reason/member plan linkage as applicable).

- A CoverageBenefit record is created/updated with Verification_Status__c, Status_Reason__c, and API_Response_Ref__c.
- The Integration_Log__c records the inbound payload and any platform exceptions with Status_Code__c.

Tip: Use distinct values for API_Response_Reference__c to make searching and troubleshooting easier in logs.

---

7) Sample Request & Response Formats

Use this to simulate results in lower environments; it updates CBVR and drives CoverageBenefit creation.


```
{
 "CareBenefitVerifyRequestId": "0n2fn0000000HvJAAU",
 "Name": "testReq2",
 "Verification_Status__c": "Acknowledged",
 "Status_Reason__c": "All details verified",
 "MemberPlanId": "0Sqfn0000000HSHCA2",
 "API_Response_Reference__c": "test"
}
```


---

8) Validation Checklist

- Permission set assigned to test user.
- External/Named Credential created and points to correct endpoint/identity.
- Queues exist and are referenced by routing (if using work items).
- CBVR creation works (UI/Flow/Apex); payload captured in Request_Payload__c.
- Integration Logs show outbound and inbound attempts with status codes.
- CoverageBenefit record is created/updated upon successful callback with correct Verification_Status__c and reference.

---

9) Troubleshooting

- No CoverageBenefit created: Verify the inbound payload hit your REST endpoint and that CareBenefitVerifyRequestId is valid; check Integration_Log__c for non-200 status or exceptions.
- Queue routing not working: Ensure CBVR_Work_Item__c creation logic is active and queues exist; confirm Benefit_Type__c matches queue routing rules.

- User can’t access objects: Re-assign Benefit_Verification_Permissions or extend FLS/permissions per your security model.