Prepared by: Protocol Layer Team

Date: 08/07/2025

Beckn Protocol Prototype Report

Purpose: Understanding current structure, flow, and proposed improvements

Introduction:

This document provides a clear overview of the Beckn Protocol prototype developed using Node.js. The prototype demonstrates how a buyer and seller can interact using standardized Beckn messages, similar to how platforms operate in the ONDC network.

Our focus as the protocol team is not on user interfaces or fixing code-level issues, but on ensuring smooth message flows, protocol compliance, and identifying ways to improve the system — such as using blockchain for better transparency.

Project Structure Overview

The prototype is organized into the following components:
BAP – Buyer App Platform: Sends requests (/search, /order, /confirm); manages user session and cart data using Redis; key file: search.js.
BPP – Beckn Provider Platform: Handles callbacks (/on_search, /on_order, /on_confirm); provides catalog, payment, and fulfillment services; key files: on_search.js, catalog.js, Order.js.
Redis: Stores temporary cart and session data.
Client (Frontend): React-based UI; not handled by the protocol team.
Gateway & Registry: Simulate discovery and message routing between apps.

How the Protocol Flow Works

This prototype follows a typical Beckn transaction flow:

Search Callback: Seller responds with /on_search.
Select + Order: Buyer selects a product and sends /order.
Confirm: Buyer confirms the order using /confirm.
Callbacks: Seller replies with on_order, on_confirm, etc.

Each message is handled by specific files in BAP or BPP, enabling real-world asynchronous communication between buyer and seller platforms.

Suggested Enhancements (Protocol-Focused)

While development improvements are in progress, our focus remains on making the protocol layer more robust and future-ready. Below are our key suggestions:

Improvements:

- Add proper logging and tracing:
  To track request IDs and message flows clearly in logs.
- Check protocol version compatibility:
  Accept only supported core_version values to avoid mismatches.
- Validate environment variables at startup:
  Ensure required .env values are present before running the app.
- Use Redis efficiently:
  Apply expiration to session keys and remove unused data automatically.
- Add API documentation (Swagger/OpenAPI):
  Provide clear specifications of available endpoints and data formats.

Proposed Blockchain Integration

We suggest using blockchain to log key protocol events (like order confirmation) for transparency, traceability, and tamper-proof auditing.

Why Blockchain?

• Ensures data is secure and unchangeable
• Builds trust between platforms
• Creates a permanent history of transactions

How It Could Work:

• Deploy a smart contract (Ethereum or Polygon testnet) to log:
  – logSearch(buyerId, query)
  – logOrder(orderId)
  – logConfirm(orderId)
• Store large data on IPFS, save only the hash on blockchain
• Use web3.js in BAP/BPP to send logs automatically

This helps record every important event in a secure and verifiable way.

Conclusion and Next Steps

The prototype shows that a basic Beckn-based transaction works as expected. Our focus ahead is to improve:

- **Message tracking and logging**
- **Protocol rule enforcement**
- **Transparent blockchain recordkeeping**
- **Technical documentation**

Next Steps:
• Test the full transaction cycle (search → order → confirm)
• Add version checks and logging
• Try blockchain logging on testnet
• Prepare protocol documentation and message flows

This report explains how the protocol layer works, what's implemented, and what we plan to improve