**QNNEX Prototype – Technical & Integration Documentation**

Date: August 2025

Version: 1.2.0

1. **Project Overview**

QNNEX is a modular, BECKN-protocol-based network for e-commerce and logistics interoperability. It enables seamless, standardized communication between buyer and seller platforms via a central Gateway, ensuring discoverability, transaction management, and real-time updates.

1. **System Architecture**

**Components:**

* **Buyer App Platform (BAP):**  
  Handles buyer-side actions (search, select, order, status, confirm, cancel) and receives callbacks for protocol events.
* **Business Platform Provider (BBP):**  
  Handles seller-side logic (catalog, order acceptance, status updates, confirmations, cancellations).
* **Gateway:**  
  Central routing and protocol enforcement layer. Receives requests from BAP, forwards to BBP, and vice versa. Manages sessions, protocol validation, and routing.

**Data Flow:**

1. Buyer initiates a search via BAP.
2. BAP sends the request to Gateway.
3. Gateway routes the request to BBP.
4. BBP responds with catalog data.
5. Buyer selects items and places an order.
6. Gateway routes order and confirmation messages.
7. Status updates and cancellations are handled asynchronously.
8. **BECKN Protocol Specifications**

**Message Structure:**

* context: Metadata for routing and identification.
* message: Payload for the specific action (search, order, etc.).

**Sample Context:**

{

"domain": "nic2004:60232",

"country": "PKR",

"city": "std:080",

"core\_version": "0.9.3",

"transaction\_id": "1234567890",

"message\_id": "abcdef123456",

"timestamp": "2025-08-06T12:00:00Z",

"bap\_id": "buyer-app.qnnex.com",

"bpp\_id": "seller-app.qnnex.com"

}

**Sample Message (/search):**

{

"intent": {

"item": {

"descriptor": { "name": "apples" }

},

"fulfillment": {

"end": { "location": { "gps": "12.9716,77.5946" } }

}

}

}

1. **API Endpoints**

**Gateway Endpoints**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /search | POST | Route search to BBP |
| /on\_search | POST | Route search response to BAP |
| /select | POST | Route select to BBP |
| /on\_select | POST | Route select response to BAP |
| /order | POST | Route order to BBP |
| /on\_order | POST | Route order response to BAP |
| /status | POST | Route status to BBP |
| /on\_status | POST | Route status response to BAP |
| /confirm | POST | Route confirm to BBP |
| /on\_confirm | POST | Route confirm response to BAP |
| /cancel | POST | Route cancel to BBP |
| /on\_cancel | POST | Route cancel response to BAP |
| /health | GET | Health check |
| / | GET | Welcome message |

**BAP Endpoints**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /search | POST | Buyer initiates search |
| /select | POST | Buyer selects item(s) |
| /order | POST | Buyer places order |
| /status | POST | Buyer checks order status |
| /confirm | POST | Buyer confirms order |
| /cancel | POST | Buyer cancels order |
| /on\_search | POST | Callback for search result |
| /on\_select | POST | Callback for select |
| /on\_order | POST | Callback for order |
| /on\_status | POST | Callback for status |
| /on\_confirm | POST | Callback for confirm |
| /on\_cancel | POST | Callback for cancel |

**BBP Endpoints**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /on\_search | POST | Respond to search |
| /on\_select | POST | Respond to select |
| /on\_order | POST | Respond to order |
| /on\_status | POST | Respond to status |
| /on\_confirm | POST | Respond to confirm |
| /on\_cancel | POST | Respond to cancel |

1. **Integration Rules & Steps for External Buyer/Seller Platforms**

**General Rules**

* **Use JSON:** All requests/responses must be JSON with Content-Type: application/json.
* **Context Block:** Every request must include a valid context object.
* **Unique IDs:** Use unique transaction\_id and message\_id for each transaction.
* **Callbacks:** Handle all /on\_\* endpoints asynchronously.
* **Validation:** Validate payloads against BECKN schemas before sending/receiving.
* **Logging:** Log all request/response cycles for traceability.

**Buyer Platform Integration Steps**

1. **API Calls:**

* Implement HTTP POST requests to BAP endpoints (/search, /select, /order, etc.).
* Include the required context and message blocks.

1. **Session Management:**

* Support session tokens/cookies if enabled.
* Track cart and order state across requests.

1. **Handle Callbacks:**

* Listen for /on\_search, /on\_order, /on\_status, etc. from Gateway/BAP.

1. **Error Handling:**

* Handle NACKs, timeouts, and protocol errors gracefully.

1. **Security:**

* Use HTTPS for all API calls.
* Implement authentication if required.

**Seller Platform Integration Steps**

1. **API Calls:**

* Implement HTTP POST requests to BBP endpoints (/on\_search, /on\_order, /on\_status, etc.).
* Include the required context and message blocks.

1. **Inventory & Order Management:**

* Keep inventory data up to date.
* Accept/reject orders, update status, handle cancellations.

1. **Handle Callbacks:**

* Listen for order and status updates from Gateway/BBP.

1. **Error Handling:**

* Respond with appropriate error codes/messages for invalid requests.

1. **Security:**

* Use HTTPS for all API calls.
* Implement authentication if required.

1. **System Requirements**

**Registration:**

* Email verification mandatory; mobile optional/recommended.
* Strong password policies.
* Prevent duplicate registrations.
* Secure session management.

**Order Management:**

* Cart and checkout flow.
* Real-time inventory checks.
* Order edit/cancel window.
* Capture cancellation reasons.

**System Features:**

* Live order tracking.
* Webhooks for payment updates.
* Notifications for critical events.
* Seller dashboards and admin controls.
* All data secured via HTTPS/encryption.

1. **Technology Stack**

* Node.js + Express: Backend APIs.
* Redis: Session/state management.
* MongoDB: Catalog and order records.
* PostgreSQL: Logging and transactions.
* Kafka/NATS: Async messaging.
* Axios: HTTP client for inter-service calls.
* Security: HMAC/RSA signing, HTTPS.
* BECKN JSON Schema: Protocol validation.

1. **Sample Payloads**

**/search (Buyer → BAP):**

{

"context": { /\* see above \*/ },

"message": {

"intent": {

"item": { "descriptor": { "name": "apples" } }

}

}

}

**/on\_search (BBP → Gateway/BAP):**

{

"context": { /\* see above, with bpp\_id set \*/ },

"message": {

"catalog": [

{

"id": "item-1",

"descriptor": { "name": "apples" },

"price": { "currency": "INR", "value": "100" },

"available": true

}

]

}

}

**/order (Buyer → BAP):**

{

"context": { /\* see above \*/ },

"message": {

"order": {

"item\_id": "item-1",

"quantity": 2,

"fulfillment": { "end": { "location": { "gps": "12.9716,77.5946" } } }

}

}

}

**/on\_order (BBP → Gateway/BAP):**

{

"context": { /\* see above, with bpp\_id set \*/ },

"message": {

"order": {

"id": "order-123",

"status": "accepted",

"items": [{ "id": "item-1", "quantity": 2 }]

}

}

}

1. **References**

* [BECKN Protocol Spec](vscode-file://vscode-app/c:/Users/DELL/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* QNNEX Prototype README