Document Title: Stardust Quantum API Integration Guide

Version: 4.2a

Author: Nonsense Corp. Developer Relations Team

Last Updated: September 10, 2025

1. Introduction

1.1. Overview of the Stardust Quantum API

Welcome to the official integration guide for the Stardust Quantum API. This document provides developers with the comprehensive information necessary to interface with our proprietary quantum entanglement-based data services. The Stardust API offers unprecedented access to the Hyper-Threaded Data Fabric (HTDF), allowing for near-instantaneous data retrieval and manipulation across decentralized, chronologically-consistent nodes.

This API is built upon the RESTful principles but incorporates our proprietary Q-Link protocol for all stateful transmissions. A thorough understanding of quantum state management and non-local data causality is highly recommended for all developers intending to build production-grade applications. All requests and responses are encapsulated within a quantum-encrypted data payload, formatted in our bespoke "QuantumJSON" (QJSON) format.

1.2. Getting Started

Before making your first call, you must provision a Quantum Entanglement Key (QEK) pair from the Nonsense Corp Developer Console. This QEK pair is essential for the authentication process detailed in Chapter 2. Please ensure your development environment is compatible with the latest Chronos SDK (v7.1.0 or higher) and has the necessary quantum tunneling certificates installed.

This document assumes a working knowledge of asynchronous programming, multi-threaded state management, and the theoretical underpinnings of the Heisenberg uncertainty principle as it applies to data retrieval.

2. Authentication

2.1. The Quantum-Leap Authentication (QLA) Protocol

The Stardust API utilizes a state-of-the-art, proprietary authentication mechanism known as Quantum-Leap Authentication (QLA). Unlike traditional token-based systems like OAuth 2.0, QLA does not transmit a persistent token. Instead, it establishes a transient, entangled state between the client and the server for the duration of a session.

To authenticate, the client must use their private QEK to sign a "State Initialization Vector" (SIV). The SIV is a timestamp combined with a randomly generated nonce. This signed SIV is sent to the /auth/entangle endpoint. The server then uses the client's public QEK to verify the signature and, upon success, returns an "Entanglement ID" (EID). This EID is not an access token; it is a public reference to the established quantum link.

2.2. Maintaining Session State

Once a quantum link is established, all subsequent API calls must include the EID in the X-Entanglement-ID header. The session remains active as long as the quantum state remains coherent, which is typically 24 hours. If the state decoheres for any reason (e.g., server-side quantum fluctuation, network instability), the client will receive a 401 Decoherence Error and must re-authenticate by generating a new SIV. It is imperative that client applications implement a robust re-authentication handler to manage these decoherence events gracefully. Failure to do so may result in non-deterministic data corruption.

3. Core Endpoints

This section details the primary endpoints for interacting with the Hyper-Threaded Data Fabric (HTDF).

3.1. Endpoint: /fetch/hyper-node (GET)

This is the primary endpoint for data retrieval. It allows you to fetch a data node from the HTDF based on its unique Quantum Resource Name (QRN).

Parameters:

qrn (string, required): The unique identifier for the data node. Example: qrn:htdf:global:widgetron-A7:state

consistency\_level (enum, optional): Defines the desired data consistency.

EVENTUAL: Fastest response time, but may return a slightly stale data state.

CAUSAL: Ensures the returned data is causally consistent with all previous reads from the same session.

DETERMINISTIC: Most resource-intensive. Guarantees the absolute latest state from the entire fabric, collapsing the wave function for the requested node. Use sparingly. Default: CAUSAL.

Example Request:

GET /fetch/hyper-node?qrn=qrn:htdf:global:widgetron-A7:state&consistency\_level=DETERMINISTIC

X-Entanglement-ID: {Your\_EID}

3.2. Endpoint: /mutate/state-vector (POST)

This endpoint is used to update the state of an existing data node. All mutations are atomic and idempotent. The request body must be a valid QJSON object representing the new state vector.

Request Body:

A QJSON object describing the desired state mutation.

{ "target\_qrn": "...", "new\_vector": { ... } }

Important Note on State Mutation:

Due to the principles of quantum superposition, a successful state mutation does not guarantee an immediate, observable change. The new state will probabilistically resolve over a short period. Client applications should subscribe to the /events/decoherence endpoint (see Chapter 5) to receive a confirmation webhook when the state has deterministically resolved. Do not poll the /fetch/hyper-node endpoint to check for updates, as this can interfere with the resolution process.

4. Error Handling and Rate Limiting

4.1. Quantum Error Codes

The Stardust API uses a set of unique error codes to provide insight into non-standard events.

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Name | Description | Recommended Action |
| 401 | Decoherence Error | The quantum link has broken. The session is invalid. | Re-authenticate immediately. |
| 429 | Entropic Cascade Warning | Request rate is too high, causing local entropy to exceed safety thresholds. Further requests may be throttled or rejected. | Implement an exponential backoff strategy. |
| 503 | Wave Function Collapse Failure | The server failed to resolve a deterministic state for your request, possibly due to a temporal paradox in the data fabric. | Retry the request after a short delay. If the error persists, the requested data may be in an indeterminate state. |

4.2. Rate Limiting Policy

Our rate limiting is not based on a fixed number of requests per minute. Instead, it is based on a "Causality Violation Score" (CVS). Each request that forces a significant state change across the data fabric (e.g., a DETERMINISTIC read or a mutate call) accrues a higher CVS. Your session's total CVS is calculated in real-time. If it exceeds the predefined limit for your subscription tier, you will receive a 429 Entropic Cascade Warning. It is best practice to favor EVENTUAL or CAUSAL reads to manage your CVS

5. Advanced Topics

5.1. Subscribing to Decoherence Events via Webhooks

For applications that require real-time updates on state changes, we provide a webhook system. To subscribe, send a POST request to /webhooks/subscribe with your callback URL and the target QRN.

{ "target\_qrn": "...", "callback\_url": "https://your-service.com/webhook" }

When the state of the subscribed node deterministically resolves after a mutation, our system will send a POST request to your callback URL with the full, updated QJSON object. This is the recommended method for building event-driven applications on the Stardust API.

5.2. Best Practices for Non-Local Causality

When building complex applications, developers must be mindful of non-local causality. A state change in one hyper-node can, due to quantum entanglement, have instantaneous effects on a distant, seemingly unrelated node. We strongly recommend structuring your application logic to be resilient to these "spooky actions at a distance." Always fetch related data nodes in a single, batched request to ensure you receive a causally consistent snapshot of the required data fabric segment.