OneChain Technology Specification

# Introduction

This document presents a detailed view of the OneChain technology infrastructure. In developing the OneChain technology stack, we were working to solve several problems, but there is one problem in particular that we addressed:

Problem: Each cryptocurrency exchange, futures execution platform, data feed provider, etc. has their own unique API and connection specifications.

Solution: By developing a single, proprietary API, OneChain can rapidly develop client apps that interface to many exchanges and other provider APIs.

# Overview

Let’s begin with a high-level diagram of the OneChain technology infrastructure:

<diagram here>

This diagram shows how various OneChain client apps (trading GUI, automated trade algorithms, aggregator) utilize the OneChain API (“OCAPI”) to provide a translation layer to specific exchanges and data providers.

# OneChain API Detailed View

If we zoom in on the OCAPI, we can get a better look at *how* the OneChain technology stack (and OCAPI in particular) facilitate OneChain software development:

<diagram here>

The diagram above demonstrates in detail how all OneChain apps are able to interact with many different exchanges, trading APIs, and data feeds. By using the OCAPI rather than the individual provider APIs, OneChain apps can ignore the myriad differences and unique specifications introduced by the individual APIs.

At a basic level, any OneChain app (trading GUI, automated trade algorithms, etc.) need only deal with the streamlined interfaces and objects exposed by OCAPI:

### Public

* Ticker – current bid/ask/last in each market
* Book – order book of bids/offers, each with an amount at price
* Trade – trade that has occurred within a specific market (*not* specific to our user)

### Private

* Order Manager – interface that provides the ability to submit new orders and cancel existing orders
* Order – one or more user orders and their current state (active, filled, cancelled, etc.)
* Trade – a trade that has been generated by filling one or more user orders
* Balance = latest account balance in each currency or product

You will notice that the OCAPI functionality is split into “public” and “private” sections. In general, private OCAPI functionality requires the appropriate API key and credentials for each exchange. These credentials are maintained by the Security Manager component, and these API keys reside on encrypted, *local* storage. Maintaining this information locally enhances security of API keys. These keys can even reside on cold storage (a memory stick, for example, that is only inserted when launching a OneChain app).

At a similar level to the Security Manager is the OCAPI Performance Monitor component. The Performance Monitor is responsible for monitoring the health of individual exchange (API) connections. This monitoring includes recognizing any interruption in exchange connectivity in addition to measuring the speed of exchange network responses (and at a higher level, how often we are able to execute trades at our desired prices).

# Provider Connections

As described above, the primary benefit of OCAPI and the entire OneChain technology infrastructure is providing a consistent interface to any OneChain client applications. This is achieved by translating from the consistent OCAPI functions to the specific functionality exposed by each exchange, trading platform, data feed, or other provider API.

There is also another, lower level at which the OneChain infrastructure operates: connection type. The majority of provider APIs utilize one of three general connection types:

1. REST – request and response via http/https
2. WebSocket – streaming socket connection via http/https
3. FIX – Financial Interchange format (<http://www.fix.org>)

If a provider API does not use one of these three connection types, then we consider it a “custom API” connection. Custom connections may operate via standard internet protocols (i.e. TCP/IP) or they may require dedicated network access (i.e. CME Direct Market Access).

The OneChain technology infrastructure is able to handle all of the aforementioned connections. In terms of custom API connections, the OneChain infrastructure currently supports TTAPI, CTS T4 API, and DTN/IQ IQFeed API.

Many providers support multiple connection types. In these instances, the OneChain infrastructure attempts to use the connection that will provide optimal performance (while reserving secondary and tertiary connection types for failover in case of connection loss).

# OneChain Multiplexer (“MUX”)

The OneChain technology infrastructure employs a distributed architecture which allows for maximum performance, reliability, and scalability.

At the heart of this architecture is the tersely-named “MUX.” The MUX, or Multiplexer, provides the intelligence to manage multiple (potentially distributed) connections across a variety of provider APIs and share these connections with all OneChain apps:

<diagram here>

As seen in this example, the MUX can even manage multiple connections to the same provider (GDAX in this case). When the MUX has multiple connections to the same provider, it automatically initiates load-balancing and failover (performance and reliability).

The example above shows the MUX and various provider API connections running on separate servers. While this is typical of the MUX and OneChain’s distributed infrastructure architecture, it is not required. Each of these components is highly optimized and lightweight in terms of both processing power and memory footprint. In fact, the entire diagram show above—multiple provider connections, MUX, and even all four client applications—can run effectively on a single desktop/server computer.

# Summary

This document has provided an overview of the OneChain technology infrastructure and OCAPI. We have seen how this technology stack lends itself to rapid development of OneChain client apps by utilizing OCAPI. We have also seen the ability of the OneChain infrastructure to handle all major provider connection types (REST, WebSockets, FIX, and custom). And finally, we were introduced to the MUX and its ability to enhance OneChain infrastructure performance, reliability, and scalability.

By combining experience in financial markets, futures and options trading, cryptocurrencies, and high-frequency systems, OneChain has produced a technology infrastructure that accommodates existing OneChain client application development while also providing a solid base for future development of both trading and blockchain-related technologies.