Last edited: October 11th, 2017

HIP Identifier: Quilt

**Sponsors:**

Takahiro Inaba – NTT DATA (inabatk@nttdata.co.jp)

Adrian Hope-Bailie – Ripple (adrian@ripple.com)

Isaac Arruebarrena – Everis, an NTT DATA Company (isaac.arruebarrena@everis.com)

**Abstract:** We propose the inclusion of a new project focused on further development of an implementation of the Interledger protocol under the project umbrella "Hyperledger Quilt". This will include reference implementations of the necessary codecs, clients and a connector and may include development of integrations with other Hyperledger projects.

Interledger ([www.interledger.org](http://www.interledger.org/)), also known as ILP, is a protocol for making transactions across ledgers. The protocol’s standards and specifications are being defined by the open source community under the World Wide Web Consortium umbrella. The Interledger Payments Community Group currently has 263 members.<https://www.w3.org/community/interledger/>

Interledger is built on two simple primitives: **prepare** (e.g. put funds on hold) and **commit** (e.g. settle funds based on the validation of cryptographic conditions). In order to transact with Interledger, ledgers only need to agree on a cryptographic primitive (SHA256) and basic 2-phase commit semantics: Transaction Prepared, Transaction Executed and Transaction Rejected.

With ILP multi-ledger transactions are possible through **ledgers** able to prepare and commit transactions, and **connectors** that bridge ledgers by participating in the transactions on both..

In a value transfer transaction (a payment), connectors deliver funds to the beneficiary on the destination ledger in exchange for the senders’ funds on the originating ledger. Account balance updates for all parties (the sender, the receiver and the connector) take place deterministically in all ledgers (the sending ledger, the receiving ledger, and any intermediate ledger that might bridge them) when the sender executes the transfer. The sender triggers the transfer once the beneficiary has submitted the fulfilment to the receiving ledger and the connector has submitted the fulfilment to the sending ledger (The fulfilment is the cryptographically verifiable message that proves that funds have been escrowed in all the ledgers participating in an ILP transaction, and are ready to be released).

The general message sequence in an ILP payment is as follows:

1. **Sender** submits transfer request to sending ledger
2. **Sending ledger** sends notifications of transfer being prepared to sender and connector
3. **Connector** submits transfer request to destination ledger
4. **Destination ledger** sends notifications to connector and receiver
5. **Receiver** submits fulfilment to destination ledger
6. **Destination ledger** sends notifications to connector and receiver (with the fulfilment attached)
7. **Connector** submits fulfilment to sending ledger
8. **Sending ledger** *executes transfer (happens deterministically in both ledgers)* and sends notifications to connector and sender

**Context:** Ledger systems today are siloed and disconnected. Transfers of value are relatively easy within one country, or if the sender and recipient have accounts on the same network or ledger. But sending value to someone on a different network or ledger is complex and often impractical. Where connections between ledgers do exist, they are manual, slow or expensive.

Interledger is based on concepts dating back to the 1970s and 1980s, but it took the advent of Bitcoin and the global blockchain movement to make the world realize that money and value transfers could be reinvented with Internet based technologies.

Internet protocols enable information to be packetized, routed and delivered over communication networks. With Interledger, money and other forms of value can be packetized, routed and delivered over payment networks and ledgers. Interledger is designed to be used in a layered protocol architecture, similar to how the Internet protocol stack was created.

**Motivation:** The Hyperledger project is an open source collaborative effort created to advance cross-industry distributed ledger technologies. It is a global collaboration including leaders in finance, banking, the IoT, supply chain, manufacturing and technology.

Interledger forms an open suite of protocols that can enable interoperability of Hyperledger members’ ledgers: from the private ledgers of financial institutions, or mobile wallets provided by FinTech companies, to distributed ledgers and blockchains.

Currently there are two main Interledger Protocol implementations under development:

· The JavaScript Implementation (interledger.js) being developed under the [JS Foundation](https://js.foundation/). The JS Foundation, like Hyperleder, is a Linux Foundation Project<https://js.foundation/announcements/2016/10/17/Linux-Foundation-Unites-JavaScript-Community-Open-Web-Development/>

· The Java Implementation, which development at this moment is not governed from any foundation or open standards’ governance body (<https://github.com/interledger/java-ilp-core>).

We propose Hyperledger Quilt to become a project focused on development of a new implementation of the Interledger protocol. The Hyperledger Quilt components may be used within other Hyperledger Projects to get distributed atomic transaction semantics where the ledgers are deployed to track digital asset transactions (E.g. Sawtooth configured with Financial transaction family) . This will enable Hyperledger members’ distributed ledger solutions, financial institutions’ private ledgers, IoT companies’ wallets, and supply chain systems to connect with one another such that a single transaction can be performed that consists of multiple individual transactions on these different systems.

With this objective, Interledger provides:

· A set of rules for enabling ledger interoperability with basic two-phase commit semantics for distributed transactions across multiple ledgers

· A standard for a ledger-independent address format and data packet format that will enable connectors to route transaction requests

· A framework for designing higher level use-case-specific protocols

**Proposed Status:** Incubation

**Solution:** The following project workflows to be considered:

· The creation of detailed Java architectural documentation

· Development of a Java Core ILP implementation (with crypto-conditions)

· Creation of a Java ledger plugin interface document

· Development of a Java ILP Client Interface and Adapter/Plugin

· Development of a Java Standard Ledger

· Development of a standard Java Ledger API

· Development of a standard Java Connector

· Development of a Java Implementation of the Quoting Protocol

· Development of a Java based Routing Protocol (Connector-to-Connector Protocol)

· Development of the Java Simple Payments Setup Protocol

· Creation of a Java implementation of the ILP-Kit (an ILP wallet with hosted ledger and connector instances)

The aim is to create a complete Java implementation of the original protocol specification, as described at [www.interledger.org](http://www.interledger.org/)

**Effort and resources:** Everis, NTT DATA and Ripple are committing full-time engineering resources to ensure the success of this project. Many other members have already expressed interest in backing the development of Interledger’s Java implementation. The following would be the initial set of maintainers for the project:

|  |  |  |
| --- | --- | --- |
| **Resource** | **Company** | **Allocation** |
| Enrique Arizón | Everis | 70% |
| Morjan Alem | Everis | 40% |
| Vicente Pastor | Everis | 30% |
| Diogo Reis | Everis | 30% |
| Miguel Lirio | Everis | 30% |
| David Peman | Everis | 20% |
| Daniel Vilas | Everis | 30% |
| Juan Carlos Gallardo | Everis | 20% |
| Adrian Hope-Bailie | Ripple | 20% |
| Jimmie Fulton | Ripple | 10% |
| David Fuelling | Ripple | 25% |
| Andrew Gates | Applied Payments | Dependant on Availability |

This team will seek to work with the other Hyperledger projects in order to find ways to enable ledger interoperability across Hyperledger’s DLT solutions and institutions’ centralised ledgers.

In addition, other engineers from NTT DATA, Everis and Ripple will also contribute to the project over time. Members of the Interledger Payments Community Group have also shown interest in contributing to the development of this ILP implementation.

**How to:** We propose the creation of the following repositories on GitHub to manage Interledger-Java resources:

· github.com/hyperledger/interledger.java-core

· github.com/hyperledger/interledger.java-ledger

· github.com/hyperledger/Interledger.java-connector

· github.com/hyperledger/Interledger.java-cli