INTEGRA

Integra Ledger Draft Architecture

V 0.08
8/27/17

David Berger
CTO Integra, Inc.

Revision History

V 0.08	Initial Draft	8/27/17	David Berger David Fisher Matt Heck Dazza Greenwood

Table of Contents

Overview

The Integra ledger is a technology based infrastructure designed to act as a global platform to mitigate the friction and enormous inefficiencies prevalent across the landscape of legal practice. It will serve as a foundation upon which the legal and development communities can enhance their current systems and build a range of new legal applications and services that allow for unprecedented interoperability within existing business and enable an entirely new class of distributed applications and code driven law.

At its core is a global utility providing unique shareable Legal Matter identities that will act as digital glue tying together the spaghetti of disconnected systems. In Internet parlance, Integra will provide a Legal Matter Identity as a Service, backed by the block chain, leveraging a network of trust between preeminent law firms, leading companies, and major universities.

The digital gurus and prophets across all sectors of business are declaring the blockchain as the next big thing which has triggered a tipping point in the legal industry in which the technology decision makers are rapidly moving toward collective acceptance. This is a key as its general utility is a function of the extent of universal adoption. Ultimately the Integra ledger is driven by trust, not only in the technology itself but also in the participants themselves. The legal industry has a tremendous advantage in this arena. For example, every lawyer is an officer of the court providing a built-in mechanism guaranteeing a high-level of enforcement.

While the more visible blockchains such as Bitcoin focus on anonymity, i.e. the lack of trust, Integra can leverage a consortium (Global Legal Blockchain Consortium, GLBC) of the leading law firms and legal departments of Fortune 500 companies to safeguard the integrity of the technology choice and the data itself. Typical of enterprise blockchains, the requirement of specifying a specific group at the root of the trust chain dictates a private or permissioned solution as opposed to an open public solution.

Bitcoin has disqualified itself as a solution both for technological reasons such as low transaction rate and its inability to store complex data and for its unfortunate association with criminal activity and the unfathomable complex concept of mining and cyber currency.

Ethereum, the second most publically visible technology option is a potential choice but is also designed as a public network and has performance issues as well as the DAO incident, a $50 million hiccup that is too complex to explain away to a non-technologist. The Ethereum roadmap looks to eventually address most of the issues, but currently it is not an optimal choice.

Bitcoin, Ethereum, as well as most of the other alternatives also suffer from a high transaction cost associated with mining and other mechanisms geared toward establishing trust, especially when they employ proof of work consensus mechanisms fueled by cyber tokens. As Integra has a high level of implicit trust due to the participants, we turned to the tokenless Linux Foundation Hyperledger Fabric blockchain as well as other Hyperledger projects including Composer and Indy.

Hyperledger provides a technology backbone acting as a foundation for building blockchain applications. Fabric promotes its main features as

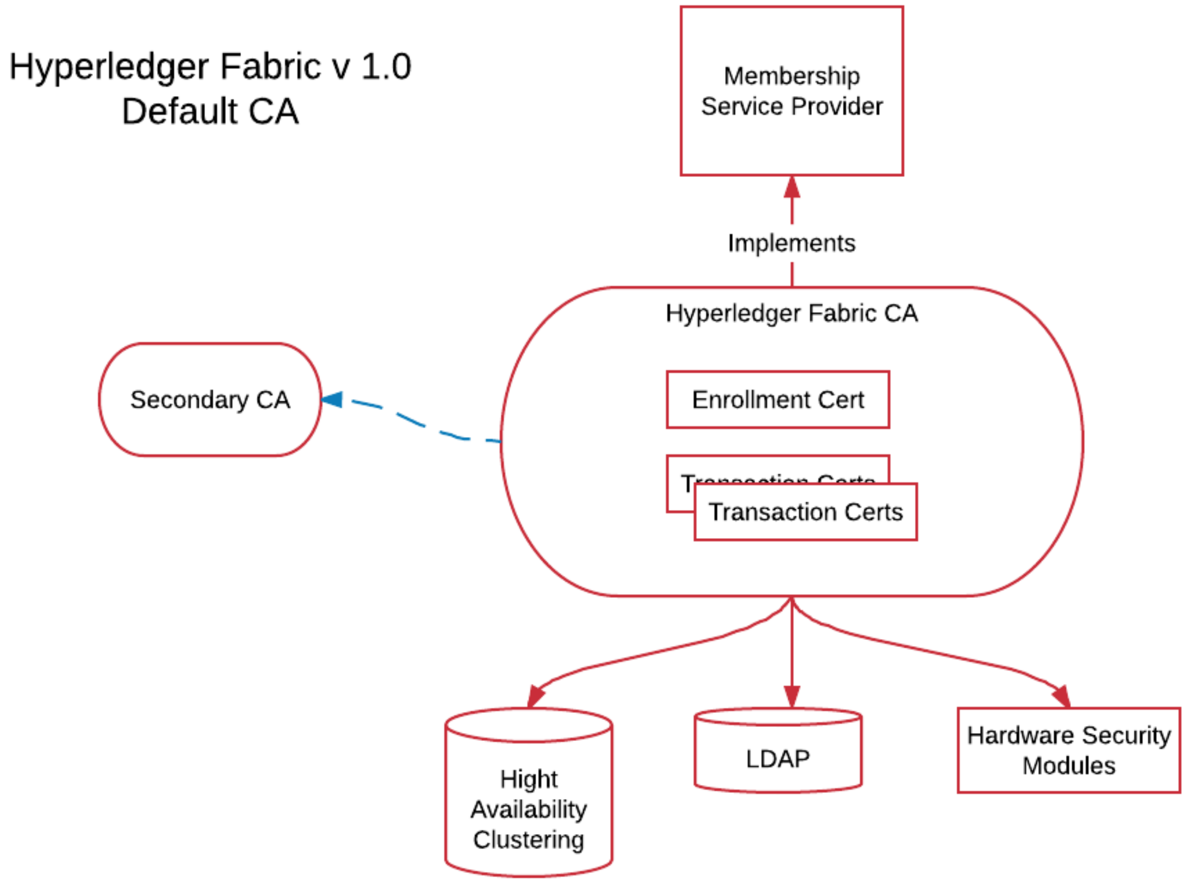
- A shared ledger/database protected by modern encryption and cryptography
- A smart contract “Chaincode” engine
- Enhanced privacy and permission enforced through membership services
- A modular architecture where all major features are designed as pluggable components

Trust Network

The architecture of the Integra ledger is centered around a network of trust designed to be interoperable with the major established trust frameworks. A key factor is trust in the encryption technology itself and the immutability of the shared ledger as information is shared with one’s competitors. Integra leverages Hyperledger’s separation of pluggable components which can be upgraded as needs changes and as more powerful and performant techniques and technologies become available. Fabric also separates trust assumptions and operation of chain code execution from ordering which provides a scalability advantage as well as an extra layer of security as different independent policies can be applied based on the identities (“trustworthiness”) of the participating nodes.

Hyperledger relies heavily on cryptography and has abstracted out this functionality to a component, the Crypto Service Provider (CSP). The CSP also supports applying different CSPs to different components of the system. As a pragmatic matter, we assume that Integra will initially launch utilizing the Hyperledger Fabric v 1.0 default crypto implementation. Due to the central role of this component we anticipate this this will be one of the first components, in a later version, where the system could gain increased efficiency and security from the upgrade to potentially newer and more powerful components.

Currently Hyperledger utilizes a PKI (public key infrastructure) with separate certificate for membership and transactions. Due to the complexities of securing and maintaining multiple certificate authorities, the CAs themselves will be maintained through proxied authority by a third party trusted agent of the consortium. Access and privileges to the ledger will be controlled through these CAs. Within the member organization, individual users can be validated through LDAP services.



One of the strengths of Hyperledger Fabric 1.0 is the support for multiple Membership Service providers. Integra will launch utilizing the Fabric built in CA SDK for MSP functionality serving the network peers (primarily law firms and corporate legal departments). Non-peer participants in the network such as individuals consuming legal services will interact though a Web API layer secured through the central CA but will likely utilize the Sovrin/Hyperledger Indy self-sovereign identity system.

Hyperledger Architecture

In Hyperledger 1.0 transaction processing is separated from transaction ordering.

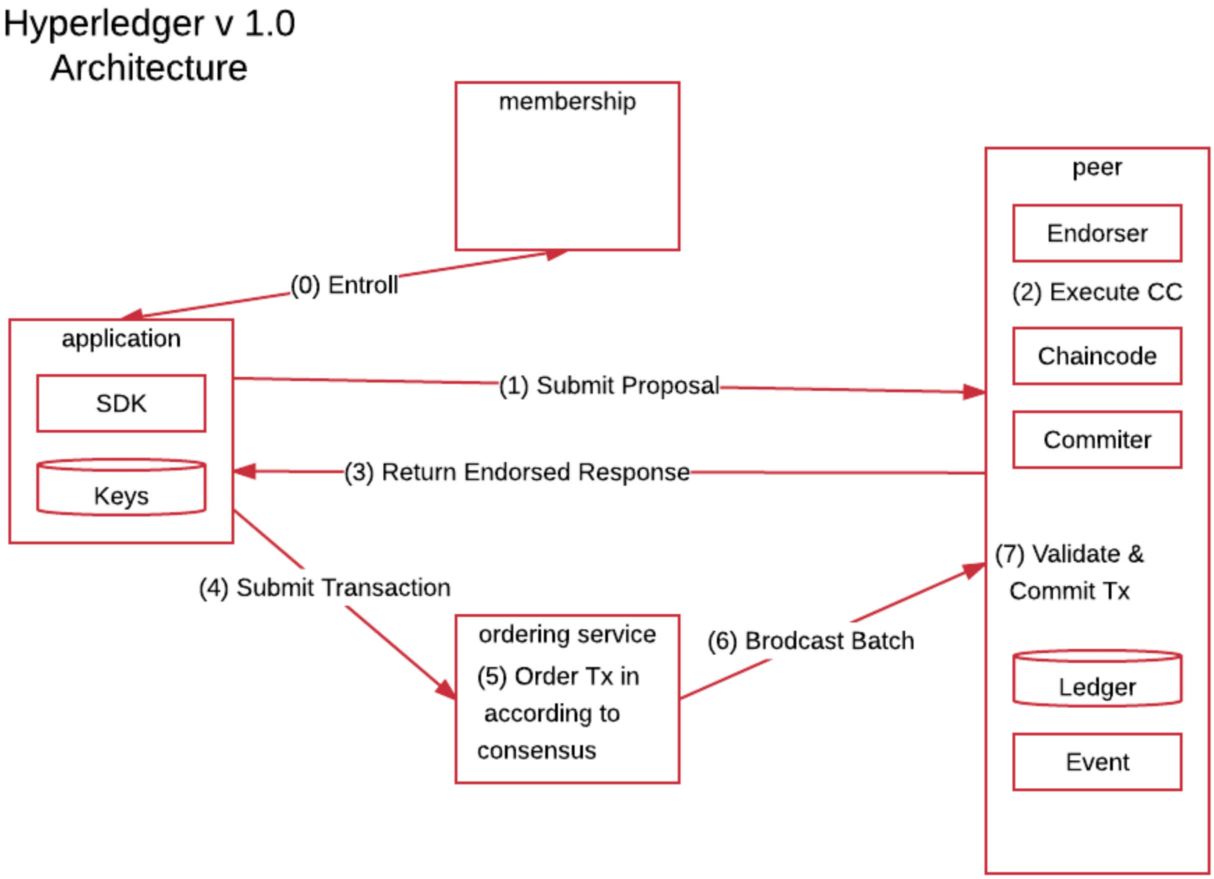
Interaction with the blockchain ledger occurs through transactions between peer nodes which are responsible for maintaining the state of the ledger as well as Chaincode. Transaction process occurs through execution of Chaincode with the **membership identity services** controlling fine-grained access privileges at the method execution level. The identities are validated again through peer issued Enrollment certificates. The access control mechanism introduces a hierarchy of peer type. In Hyperledger Fabric 1.0, peer identities can be masked using one time transaction certificates.

Endorsers executes and validates transactions through validating method payloads and executing chain code.

Committers verifies endorsements and validates transaction results. Endorsers are a sub class of committers so can act in both roles.

Orderers approve the inclusion of transaction blocks to the ledger but do not maintain a copy of the ledger or non-system level Chaincode.

Peers manage synchronization through peer to peer communication using a gossip protocol.



In the flow indicated above

0. A peer node requests and is granted an enrollment certificate from the membership service provider.

1. The application defines an endorsing policy. The peer node selects an endorser or set of endorsers that fulfill the requirement of the endorsing policy. It then wraps up the parameters for the chain code and sends this to the endorsers as a proposal.
2. The endorsers validate and execute the Chaincode.
3. If valid, the endorsers sign and return the endorsement along with the results of the executed Chaincode.
4. The original requesting node signs the endorsed package and submits it to the ordering service.
5. The ordering service endorses the transaction based upon the configured ordering policy.
6. The ordering nodes batch the transaction and broadcast the batch to the network.
7. The subscribed committing peers then commit.

Integra supports a taxonomy of application types described later in the document. Each class of application is assigned different endorsement and ordering policies depending on their needs. For the initial base use cases, both the endorsement and ordering/consensus policies can be simple, even allowing for self-endorsement, as there is not initially a big issue of double spending, ordering is not critical, and as due to the makeup of the membership, there is a much smaller incidence of malfeasance among the nodes.

There are, however, many use cases where transaction privacy, even in the presence of encrypted data on the ledger, must be maintained. Hyperledger Fabric 1.0 addresses this common business requirement with the introduction of channels.

Integra Ledger Nodes

In the Integra Ledger Network nodes belonging to organizations that are members of the consortium are granted the highest level of access due to their trust level. The nodes are called **Veritas Nodes** and are generally qualified to perform any endorsement, commitment or validation.

Nodes belonging to legal entitles that are not part of the consortium are called **Peer Nodes** and will be granted lesser access.

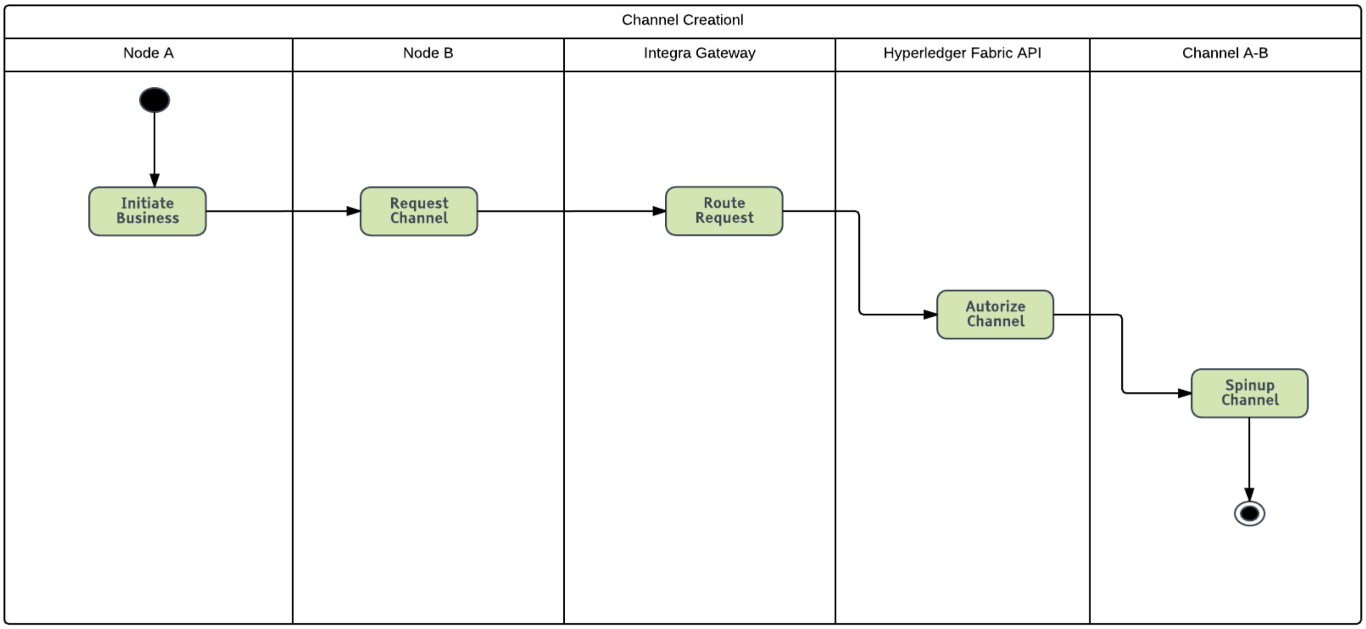
A set of nodes, named **Integra Nodes** will be operated by a trusted agent of the consortium and granted the highest level of trust available. These nodes will host the API gateways allowing access to the network for less trusted consumers.

The exact rules governing these privileges will be dictated by the consortium.

Channels

Channels are private mini side chains that contains their own ledgers, world state (NOSQL database of state changes) and Chain code execution environments. Channel standup and teardown is controlled by the Fabric API. The Integra ledger will wrap this API and provide controlled access for any system level channel operation. Channel membership consists of one or more peer nodes and can be dynamically modified after creation. Chain code is scoped to the channel level and can be shared among channels. A given node may subscribe to multiple channels. The channels execute code concurrently boosting performance and scalability. Hyperledger Fabric supports cross channel code execution, which will be extremely useful although this feature is not yet well documented.

On the surface channels appear to be a panacea for privacy requirements but members of the Hyperledger team are already warning that developers should not overly rely on channels due to the time and resources required to spin up and maintain them. In addition, channels are still evolving to meet current needs such as the planned addition of side databases for private data. We intend to support channels as a beta feature in the initial release where we hope their feasibility is valided as part of the numerous proofs of concept and experiments we encourage the community to engage in.



World State Database

The current state of the Chaincode is stored as a set of Key Value pairs. Hyperledger Fabric 1.0 supports the pluggable replacement of the data store backing the history of World States. The default option for v1.0 is CouchDB, one of the best of breed open source NOSQL databases in use today. One advantage is that CouchDB supports complex querying of the data allowing apps to perform rich functionality including:

- Keyed queries
- Composite Queries
- Key Range Queries

Integra Building Blocks

Identity

As referenced throughout this document, identity management for both individual and legal entities lies at the center of the Integra Ledger. The Ledger provides a foundational framework for building and participating in blockchain powered applications and services. The Integra Membership and Transactions services can be viewed as a backplane to a federated identity system.

Legal matters lie at the center of law practice and case management therefore needs to be treated as a first-class citizen of the Integra Ledger, of equal importance to the identity of the lawyers and firms themselves. The legal industry is embroiled in a state of chaos stemming from a haphazard patchwork of approaches toward the identification of digital legal documents and other artifacts related to legal matters. Each large law firm isolates itself within independent silos, each with its own matter management systems.

This is an extremely complex issue with no single silver bullet solution. However, we propose that a good first step in mitigating the chaos is having the Integra Ledger provide a global utility providing a matter identification registry service. Once generated, the matter ID (LMATId) can be used as a universal identifier. In other words, it would serve as a commonly agreed upon system for establishing a universal foreign key that can act as glue tying together multiple references to a single matter even when distributed between previously unconnected systems. The IDs themselves would be stored and remain addressable on an immutable trusted registry of all legal matters which would provide a single source of truth fronted by an open source standards based interface.

Sovrin (Hyperledger Indy)

Public consumption of the services will also require establishing identity. In their case, privacy and semi anonymity is critical. There are a few different solutions currently available to support this need for a non-federated identity system. As a Hyperledger based platform we will initially incorporate a Sovrin based solution as this technology has been accepted to the Hyperledger foundation as project Indy. In later versions, we intend to support a wider range of publically available identity systems. Sovrin utilizes its own blockchain targeted toward a diffuse model of trust.

From the Sovrin Website

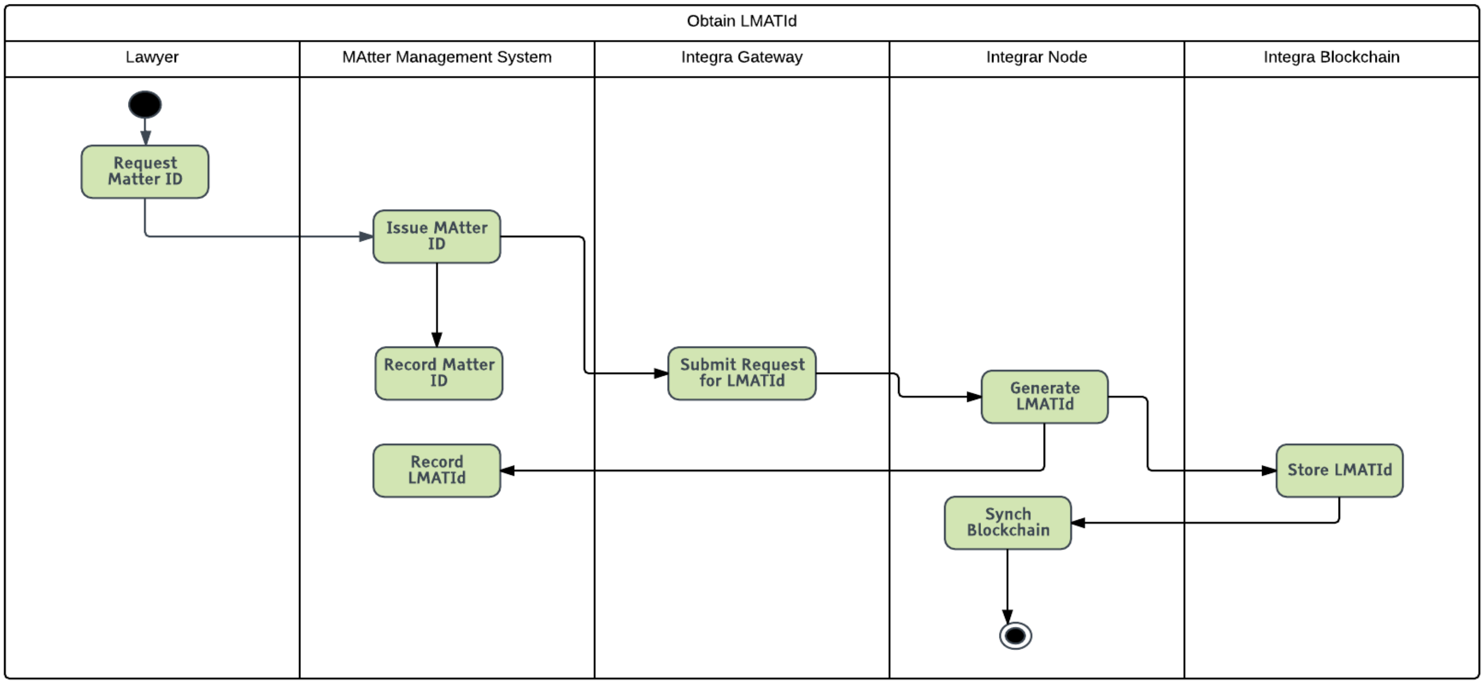
The Internet was originally designed for one thing – to allow machines to communicate with one another. As a result, machines have identities on the Internet, but people don't. This stunts communication and trust while adding enormous complexity, costing the global economy hundreds of billions of dollars each year.
Self-Sovereign Identity (SSI) is an identity that is 100% owned and controlled by an individual or organization. No one else can read it, use it, turn it off, or take it away without its owner's explicit consent. This simple form of identity is private, extremely secure, and goes where you go. This will enable trusted interactions to occur between individuals, institutions, and businesses.

Apps

As an open source development ecosystem, the Integra Ledger will allow for the creation of a myriad of applications offering new functionality and tying together a wide range of existing systems. The general tech public is focusing on the dream of complex smart contract based projects. We are currently focused on the foundation architecture but are supporting compex chain code development through the Hyperledger Fabric SDK . These will most often live on dynamically generated channels under control of the participating law firms and legal departments.

LMATId Registry (Legal Matter ID Registry)

The greatest immediate utility will be realized through universal foundational apps living on a common channel built and controlled by the consortium. We will refer to these as **foundational apps** as opposed to **private apps**. The Integra blockchain will launch with two foundational apps, the LMATId registry and a proof of concept LMAT Document Registry although we expect some of the initial POCs sponsored by consortium members will include additional foundational apps.

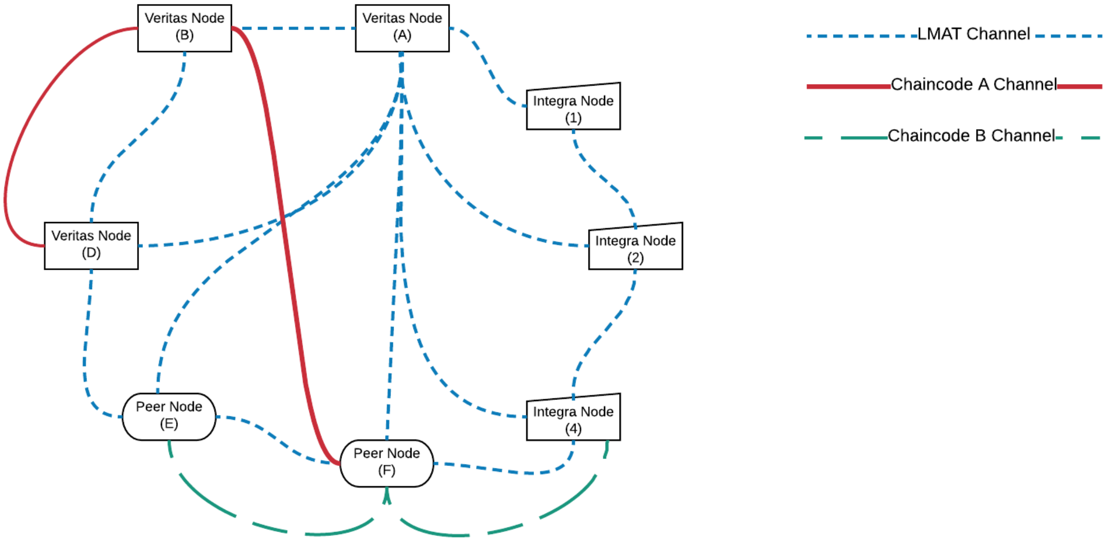


The consortium will dictate the endorsement and ordering policies of the foundational apps but will allow more freedom with private apps. The consortium will also need to validate and approve any foundational app as it can affect the entire network.

Private Apps

Private apps operate on channels which are stood up through API calls to the Integra Gateway. Chaincode is bound to the channel offering both performance and security advantages over earlier versions of Hyperledger Fabric. The ability to create channels and control channel membership is based upon permissions controlled by the MSP.

Members of the consortium are required to sponsor a proof of concept application. Many of these are likely to be private apps.



API & Data Model

By design, the model for an LMATId as well as the API layer has been kept brutally simple.

LMATId Proposed Data Model

```
/**
 * Integra Ledgra Network
* Composer CTO file
 */

namespace com.integraledger.identityregistry
asset IntegraIdentity identified by identityId {
   o String identityId
   o String identityType
   o DateTime creationDate optional
   o String value optional
   o String creatorId optional
   o String metaData optional
}
participant User identified by userId {
   o String userId
}


transaction RegisterIdentity {
  o String identityId
  o String identityType
  o String value optional
  o String creatorId optional
  o String metaData optional
}


transaction IntegraIdentityExists {
  o String identityId
}

event IdentityRegistrationEvent {
 o String identityId
}
```

API

As described in the sections relating to Hyperledger Composer, all Chaincode and data is accessed through a RESTful API exposed through Loopback. This guarantees the greatest consistency across development environments but as noted, Integra Ledger will expose direct SDK access in later revisions. We will likely support SDKs across all computer languages but for the sake of simplicity and more universal adoption we will probably present most samples in Python once the SDK becomes available.

Register Identity

Call to get an unique id. The canonical case is for an LMATID. The id type is specified in the IdentityType parameter, com.integraledger.lmat.

- **URL**
  /RegisterIdentity

- **Method:**
  POST

- **URL Params**
  None

- **Data Params**
  - identityId (String): Identity identifier such as Sovrin ID
  - identityType (String): Namespaced identity types such as com.integraledger.lmat or com.integraledger.document
  - metadata (String): Optional MetaData
  - value (String): Optional data for example for document exostance, this would hold the document hash.

```
{
  "$class":"com.integraledger.identityregistry.RegisterIdentity",
  "identityId":"6e1de38e-4c9d-4fd1-ba7e-fb65650b48f3",
  "identityType":"com.integraledger.lmat",
  "metaData":"Sample MetaData",
  "value":""
}
```

- **Success Response:**
  - **Code:** 200
  - **Content:** {

```
      "$class": "com.integraledger.identityregistry.RegisterIdentity",
      "identityId": "6e1de38e-4c9d-4fd1-ba7e-fb65650b48f1",
      "identityType": "com.integraledger.lmat",
      "value": "",
      "metaData": "Sample MetaData",
      "transactionId": "1b9f26c0-03a9-49b0-8b2f-8c7a2322dd40"
```

}

- **Error Response:**
    - **Code:** 401 UNAUTHORIZED
      Content:

  OR
    - **Code:** 422 UNPROCESSABLE ENTRY

- **Sample Call:**
```
curl --request POST \
  --url http://localhost:3000/api/RegisterIdentity \
  --header 'accept: application/json' \
  --header 'cache-control: no-cache' \
  --header 'content-type: application/json' \
  --header 'postman-token: 91631ee3-6302-4aac-c1c3-1b377b573db8' \
  --data '{  \n   "$class":"com.integraledger.identityregistry.RegisterIdentity",\n
"identityId":"6e1de38e-4c9d-4fd1-ba7e-fb65650b48f3",\n
"identityType":"com.integraledger.lmat",\n   "metaData":"Sample MetaData",\n
"value":""\n}'
```

Integra Identity

Call to get IntegraIdentity by identityId.

- **URL**
  /IdentityExists/{identityId}

- **Method:**
  GET

- **URL Params**
  None

- **Data Params**
  None

- **Success Response:**
    - **Code:** 200
    - **Content:** {
```
        "$class": "com.integraledger.identityregistry.IntegraIdentity",
        "identityId": "6e1de38e-4c9d-4fd1-ba7e-fb65650b48f1",
        "identityType": "com.integraledger.lmat",
        "value": "",
        "metaData": "Sample MetaData"
}
```
- **Error Response:**

   - **Code:** 401 UNAUTHORIZED
     Content:

  OR
   - **Code:** 422 UNPROCESSABLE ENTRY

- **Sample Call:**
```
curl -X GET \
 http://localhost:3000/api/IntegraIdentity/6e1de38e-4c9d-4fd1-ba7e-fb65650b48f1 \
-H 'accept: application/json' \
-H 'cache-control: no-cache' \
-H 'content-type: application/json' \
-H 'postman-token: d7e2e36b-b58b-f72e-fa35-fdb2f76fd28d' \
-d '[
        {
        "$class": "com.integraledger.identityregistry.IntegraIdentity",
        "identityId": "string",
        "identityType": "string",
        "creatorId": "string",
        "metaData": "string"
        }
]'
```

Integra Value Exists

Call to verify if an identity has been registered to the blockchain.

- **URL**
  /ValueExists

- **Method:**
  GET

- **URL Params**
  Id (String): the Value to check.

- **Data Params**
  None

- **Success Response:**
  - **Code:** 200
  - **Content:** {
```
    "exists": true,
    "value": "b768de9c3712cbd40d7a02e32ef061cc8c1baef0"
    }
```

- **Error Response:**

    - **Code:** 401 UNAUTHORIZED
      Content:

  OR

    - **Code:** 422 UNPROCESSABLE ENTRY

- **Sample Call:**

```
curl -X GET \
  'http://localhost:3001/api/valueExists?id=b768de9c3712cbd40d7a02e32ef061cc8c1baef0' \
  -H 'accept: application/json' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: 25952042-6b00-1d97-e0ef-b7c1396cc521' \
  -d '[
  {
    "$class": "com.integraledger.identityregistry.IntegraIdentity",
    "identityId": "string",
    "identityType": "string",
    "creatorId": "string",
    "metaData": "string"
  }
]'
```

Integra Value Exists

Call to verify if an identity has been registered to the blockchain.

- **URL**
  /ValueExists

- **Method:**
  GET

- **URL Params**
  Id (String): the Value to check.

- **Data Params**
  None

- **Success Response:**
  - **Code:** 200
  - **Content:** {
      "exists": true,
      "value": "b768de9c3712cbd40d7a02e32ef061cc8c1baef0"
      }

- **Error Response:**

- **Code:** 401 UNAUTHORIZED
  Content:

OR

- **Code:** 422 UNPROCESSABLE ENTRY

- **Sample Call:**

```
curl -X GET \
  'http://localhost:3001/api/valueExists?id=b768de9c3712cbd40d7a02e32ef061cc8c1baef0' \
  -H 'accept: application/json' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: 25952042-6b00-1d97-e0ef-b7c1396cc521' \
  -d '[
  {
    "$class": "com.integraledger.identityregistry.IntegraIdentity",
    "identityId": "string",
    "identityType": "string",
    "creatorId": "string",
    "metaData": "string"
  }
]'
```

Upcomming API

CreateChannel

Call to standup a Channel.

TearDownChannel

Call to tear down a Channel.

SetChannelPermissions

Call to set permissions on a Channel.

AddMemberToChannel

Call to add member to a Channel.

RemoveMemberFromChannel

Call to remove member from a Channel.

GetChannelInfo

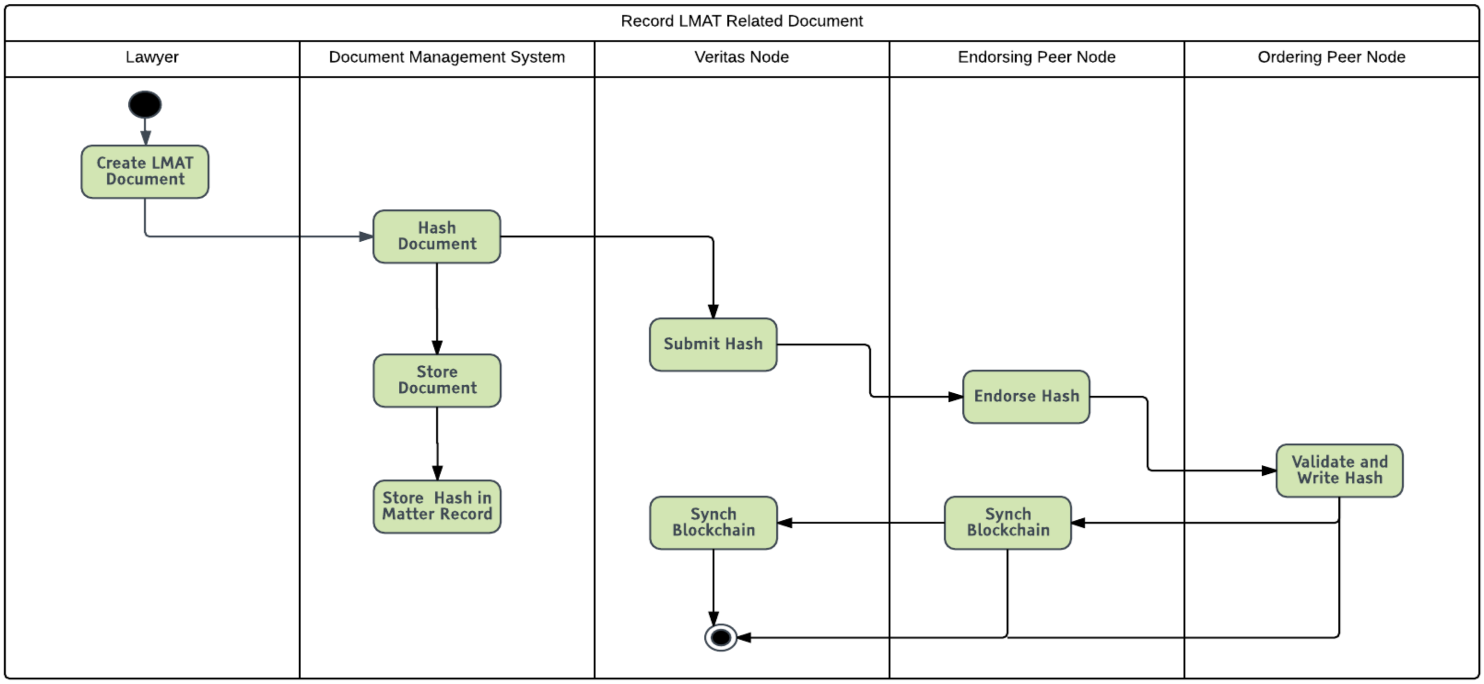
Call to return information about a Channel.

LMAT Document Registry

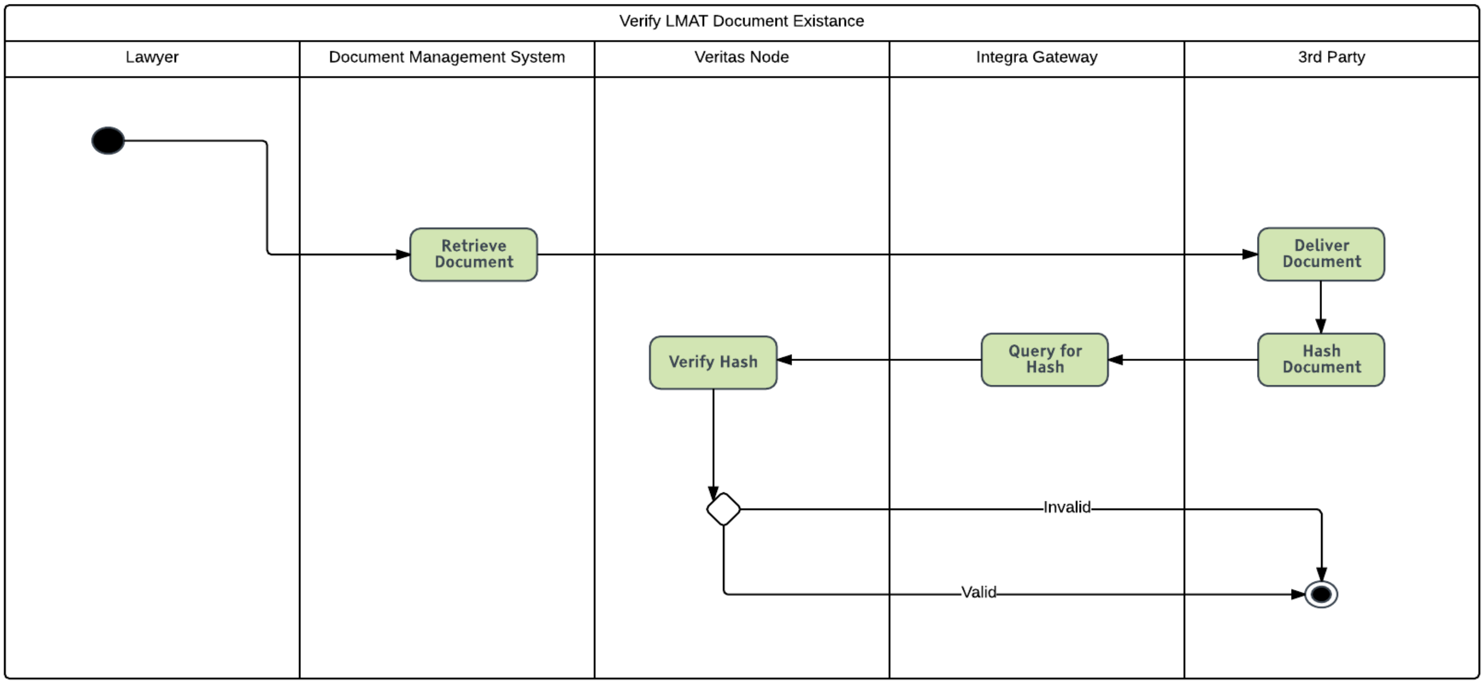
A single pdf, representing a contract between two parties associated with a legal matter may have dozens of different identifiers due to versioning and non-compatible systems from firm to firm. These systems clash with an orthogonal system used by the courts. This is further compounded by the legal industry's general resistance to technology. This in turn leads many lawyers to transfer and store documents over unsecured means such as email with non-password protected file attachments exposing private client information and leaving it vulnerable to multiple vectors of attack. At the end of a domino effect lies the compromise of these matters as you run the risk of sacrificing the ability to prove the existence or provenance of the legal matter/pdf. An LMAT Proof of Existence App provides an excellent POC/first use of the LMATId registry service.

In addition, despite the technical possibility of storing the actual binary data from the LMATs associated digital artifacts, the app will only store a SHA 3 hash of the data/file. This hash will be generated client side so the actual content of the file will never be exposed to the system or other nodes/members.

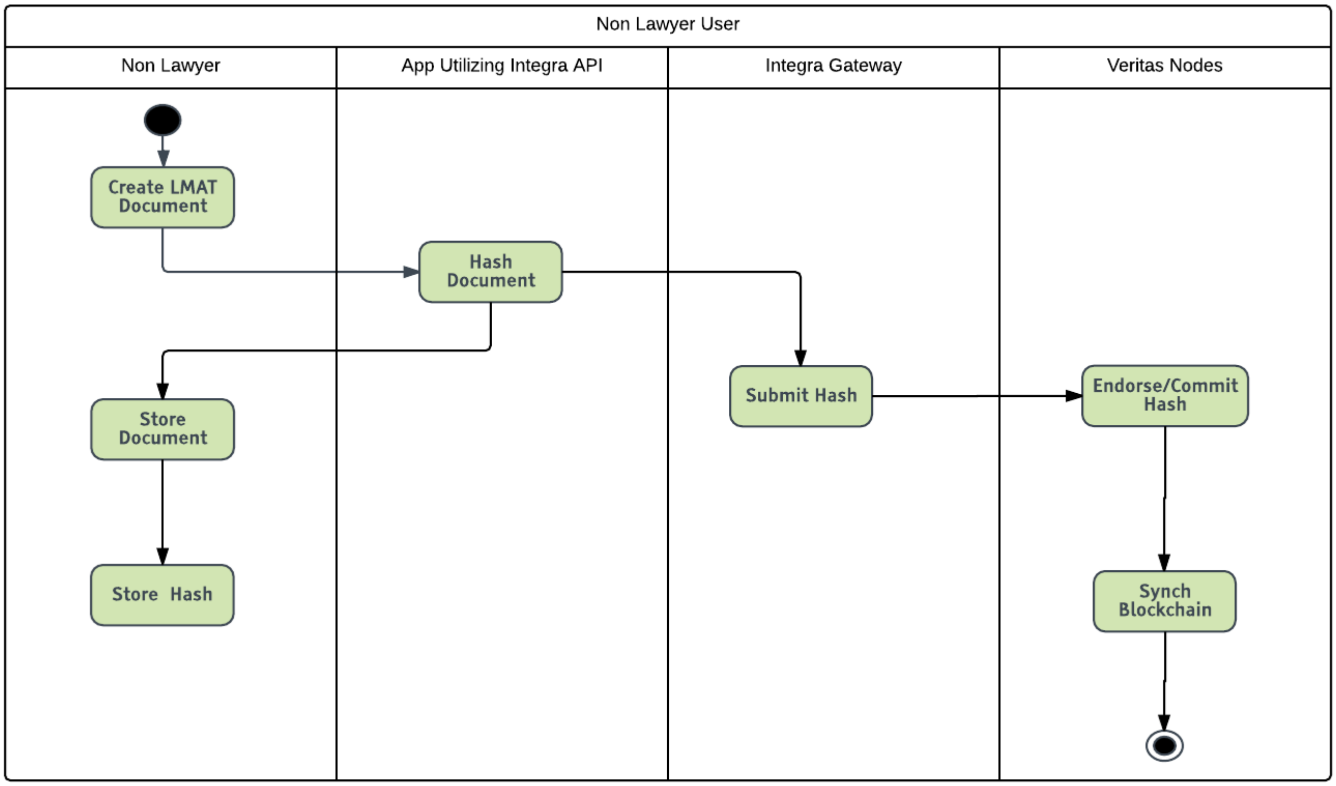
This does constrain the system in achieving one of its primary purposes, providing proof of existence. The owners of the file are responsible for securing and storing the digital representation of the LMAT/File off chain. To provide proof at a later date, one must produce the unaltered file or exact copy of the file and generate a SHA 3 hash which is then compared to the hash on the ledger. This may appear to be a burden but has been deemed an acceptable requirement by the intended audience.



Note that only the document hash and not the binary data for the document is stored on the Ledger. Integra Ledger is not a document management system, therefore an alternate solution must be employed to manage the document and subsequent revisions. This may seems like a significant constraint but we strongly believe that simply anchoring the LMATId and storing the document for later proof of existence has enormous value well worth the limitations.



As noted above, the proof of existence requires the user to retrieve the original document so it can be rehashed so that the hash can be compared against the hash stored on the ledger and returned via an Integra Gateway API call.



Design Philosophy

It is critical for the success of Integra that the initial impression and experience of interacting with the technology be as clean as possible. To help maximize this goal, we have embraced the KISS principle. From Wikipedia:

KISS is an acronym for "**Keep it simple, stupid**" as a design **principle** noted by the U.S. Navy in 1960. The **KISS principle** states that most systems work best if they are kept simple rather than made complicated; therefore simplicity should be a key goal in design and unnecessary complexity should be avoided.

One way to enact this approach is to rely heavily on design patterns and best practices as laid out by the experts. As a new technology, this consists largely of the original platform architects. One of the significant barriers for general adoption is the prevailing belief that blockchain programming is extremely hard and that a shortage of skilled domain experts will drive exorbitant development costs.

Developing Apps & Services

Hyperledger Composer

Hyperledger Composer has nicely addressed simplifying the development process meeting, helping fulfill our design goals, through the addition of the Hyperledger Composer project which is currently under incubation. We have therefore chosen Composer to provide the only supported mechanism for developing against the Integra Ledger during the initial release.

Composer provides a well-designed abstraction layer on top of the complex Fabric SDK and APIs providing tools aimed at non-programming domain experts and the vast universe of JavaScript conversant programmers. Composer describes itself as:

Really simple models

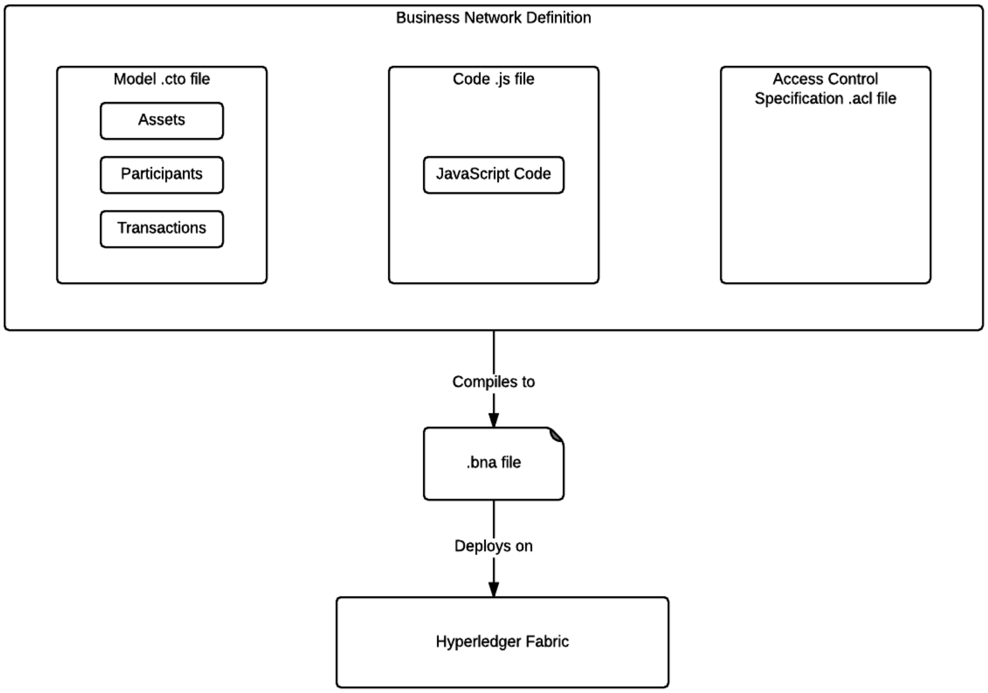
- Define a business network in our purpose-built modelling language, and script transactions in JavaScript: the most popular language on the planet.

Quick reusable POCs

- Rather than weeks, think hours to develop production-ready applications that go from the web-browser all the way back to the blockchain.

Data integration

- From enterprise systems of record to best of breed dev ops: Composer uses Loopback to connect your blockchain to your existing systems.



In Composer, the business network is developer through 3 components.

- The Model is defined using a business domain specific language designed for simplicity so that it is readable by a non-programming subject matter expert.
- Code is written in JavaScript and references objects defined in the .cto file. Composer supports an event model so that the hosting app can subscript to events related to the ledger.
- Composer simplifies the process of setting Chaincode and other permissions through support for a text based access control file, ACL.

In keeping with KISS, despite the existence of a range of language specific SDKS allowing direct access to the Fabric blockchain, we have decided to initially limit access to a RESTful API interface. This will utilize the Loopback driven data access strategy implemented by Composer. In this way, we hope that

- The use on a single development strategy will promote better reuse of examples and collaborative help among the initial users
- The single path will reduce the attack surface promoting a more secure environment
- Leverage the existing work including testing
- Increase trust and confidence as we are initially adopting the best thing provided by IBM, the originators of the platform
- Composer itself utilizes best of breed opens source components including
    - Yeoman
    - Loopback
    - Angular 2 Scaffolding

App Development

Current development requires the use of Hyperledger Composer. As a result, both new App development and current systems integration involve consuming industry standard RESTful web services which is an extremely well defined and documented development domain. Developers and systems architects will need to adjust their thinking to the new blockchain paradigms but should be able to apply their existing skills to rapidly develop both proof of concept and then production ready solutions involving the Integra Ledger.

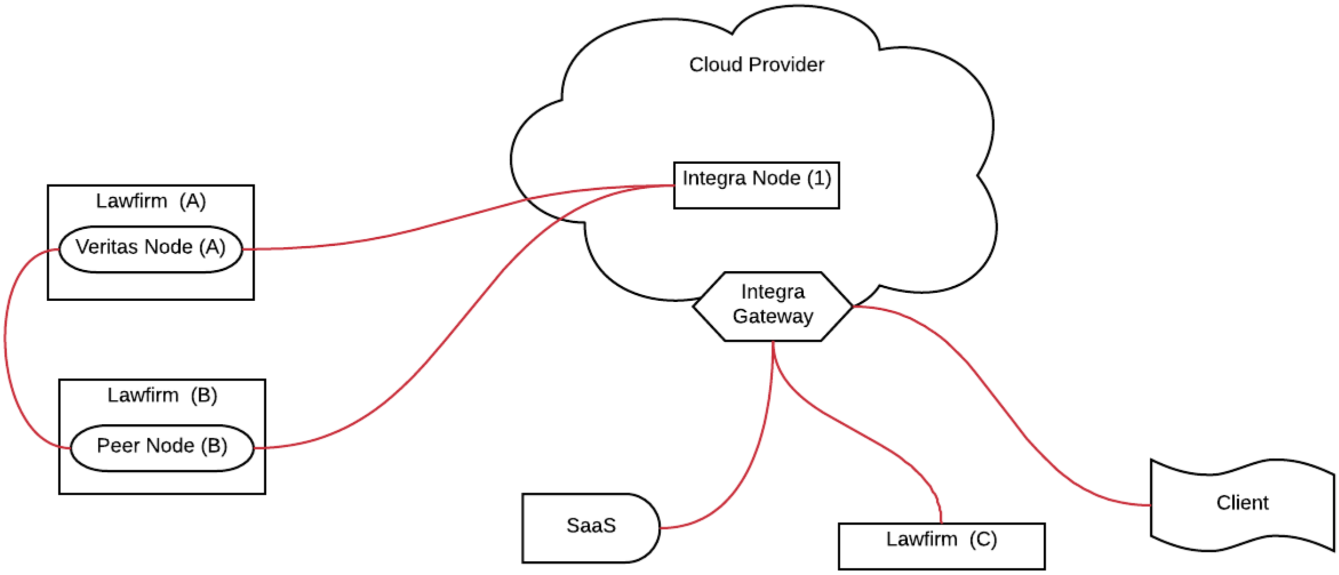
To simplify development, Hyperledger Composer provides

- An online playground, requiring no installation, that simulates a complete Composer solution using the browser's local storage to simulate the Hyperledger Fabric blockchain.
- An online coding and test environment for use in conjunction with the playground.
- Coding templates with features such as autocomplete for the open source, cross platform code editors, Atom and VSS Code.
- Docker container for easy local provisioning of a full environment.

Integra Ledger Network

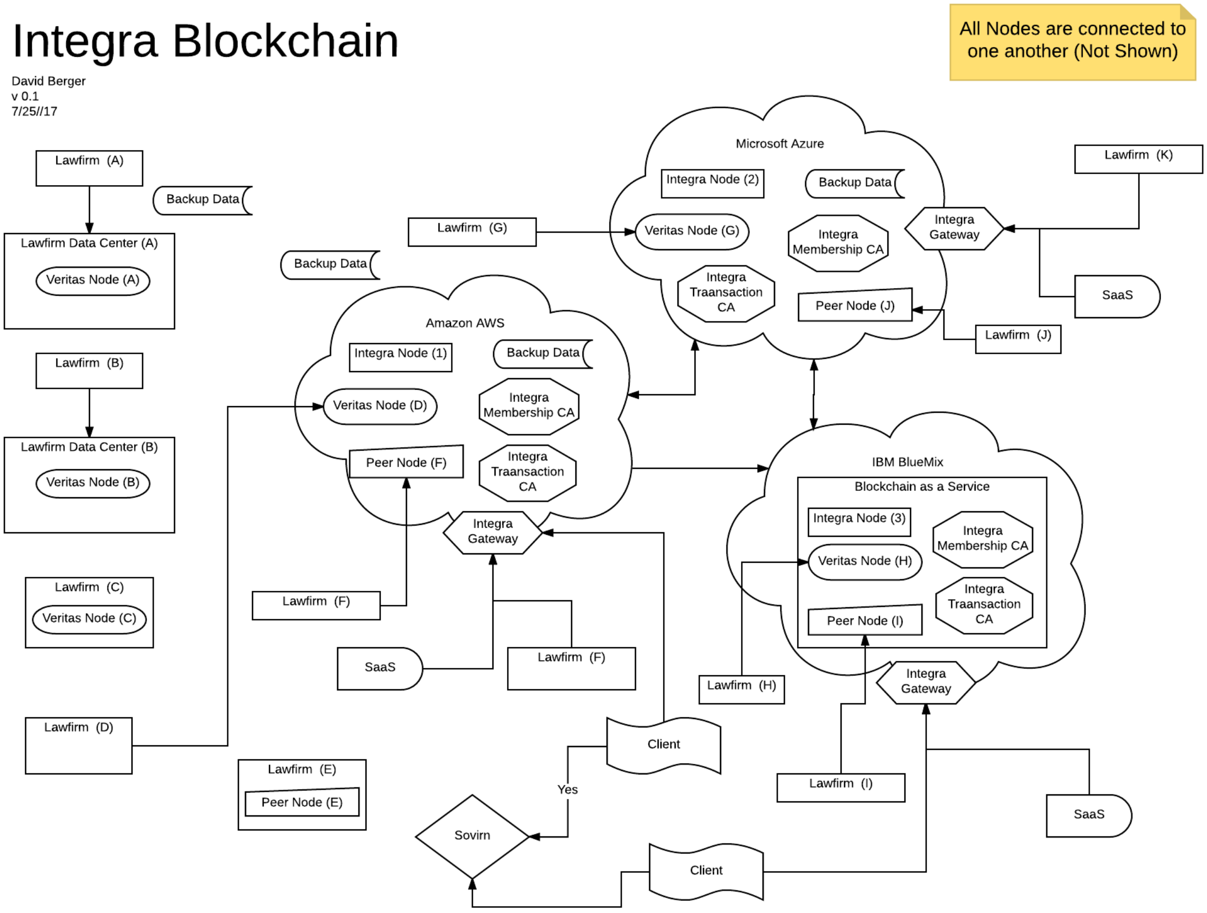
Nodes of the Integra Ledger will be provided as Docker containers allowing for easy installation both within a law firms data center or on any major cloud provider. IBM's BlueMix cloud offering supports Hyperledger Fabric 1.0 as a service so this will also be offered initially as a deployment option.

All outside consumer access including SaaS system integration, non-peer node law firms, and layman accessing member firm services will occur via RESTful API calls through a cloud hosted API named the Integra Gateway. This API will be secured through standard security on authorization mechanisms.



The Integra Nodes, hosted by an agent of the consortium will be deployed in a redundant fashion across a series of top tier cloud providers to achieve maximum fault tolerance and availability. In addition, these nodes will employ the best practice backup procedure offered by the cloud providers including automatic backup, long term backup and off site backup.

The diagram also shows the various deployment options described above as well as the CA's for the Membership and Transaction authorities and the integration of the Sovrin block chain for nonmember consumer identity verification.



Test Plan and Onboarding

Integra Ledger's software testing plan is expected to have three major components, phased in gradually as development proceeds: unit testing, use case testing, and red cell testing.
For unit testing, the individual libraries of Chaincode developed to submit changes to the Hyperledger Fabric chains will be tested against strict functional descriptions. In general, this technique results in the development of multiple tests for each individual function exposed via the API, as well as any private internal support functions where strict testing accelerates the development of the public functions. At a minimum, this means verifying that, when a function's required inputs and preconditions are met, it produces the output expected, and that when these inputs are not met, the function aborts or fails in a documented, well-understood manner. To the extent made reasonable by available technology, these tests will verify full coverage of all logical branches and code paths in the library.

When the supporting libraries are sufficiently developed, use case testing will run through various business scenarios, such as adding and revoking authorized users, creating a chain between a subset of these users, hashing a document and adding this hash to this chain, and finally, re-hashing and verifying the document against the chain. These should include both positive and negative use cases; for example, testing that an expired certificate cannot be used to execute some of the operations described. The main value of use case testing is to ensure that well-understood API components do not behave in unexpected ways when used in a full system. Finally, red cell testing involves actively running attacks against the system. While Hyperledger Fabric itself will undoubtedly receive significant and thorough security analysis, including its own red cell testing, it is still worth having a third party evaluate Integra Ledger's libraries, APIs, and actual ledger activity to try and identify any possible attack, compromise, or denial-of-service vectors above and beyond Hyperledger Fabric itself. Should such issues be identified, they can then be mitigated.

Additionally, this phase includes discussion of the possible impact of unexpected cryptanalytic developments, and a review of possible mitigation techniques, as a strategic precaution.
These three major testing strategies are well-understood by the modern software development industry, and are considered prudent and effective ways to improve product quality and reliability.

Initial User Stories

Join the Global Legal Blockchain Consortium

As a law firm, I want to join the Global Legal Blockchain Consortium so that I can contribute to the development of the standard that will drive the Integra Ledger so that my firm and the community at large will benefit from the use of blockchain technology in the legal industry.

Verify Identity of Law Firm

As a law firm, I want to verify my identity so that I can receive the certificates that enable me to conduct transactions on the Integra Ledger.

Verify Identity of Lawyer at Law Firm

As a lawyer at a law firm utilizing the Integra I want to be validated as a user through our enterprise LDAP directory so that I can conduct transactions on the Integra Ledger.

Utilize the LMATId App

As a SaaS matter management system, I want to be able to call the Integra Ledger API so that I can obtain an LMATID and associate it with my internal matter record so that there is a common identifier I can reference when communicating with systems outside my associated firm.

Store a document hash on the Integra Ledger

As a lawyer, I want to be able to store a hash of a digital document, related to a matter, on a shared immutable data source so that I can prove the existence of the original digital document by comparing a hash of the document with the time stamped copy on the Integra Ledger.

Prove Existence & Date of a Document

As a lawyer, I want to utilize the Integra Document Hash App to be able to prove the provenance of a legal document binding it to a date and time so that I can avoid disputes in court at a later date.

Create a Private Channel

As an app utilizing the Integra Ledger API, I want to be able utilize the channel creation mechanism to limit the visibility of the transactions to a subset of the nodes to meet my firm's privacy concerns as even the presence of encryption is not sufficient obfuscation.

Technology Outline

Code model

Open source where possible with the possibility of closed source if the components are pluggable and can be swapped out for an open source compatible solution.

Technology

- Block Chain
- Docker containerization
- Cloud hosting
- RESTful API access

Network Type

Provisioned

Blockchain Platform

Hyperledger Fabric

- Data access is managed through code

Network Subdivisions

- Foundation Apps
  - All nodes members
  - Under control of foundation agent
  - All transactions signed by Transaction CA under control of consortium agent
- Private Apps
  - Spawned from Chain Code factories under control of consortium agents
  - Run on dynamically generated Hyperledger Fabric channels under isolated independent chains maintaining their own world state

Membership

- Nodes utilize Membership CA. Under control of consortium agent
- Non consortium member Nodes need identity establish either
  - though standard membership service
  - Sovrin

State Management

CouchDB for Hyperledger Fabric World State management

Endorsement Policy

- Foundational App
  - Must be endorsed by all parties although this can be a single person

- Private App
    - Set by members through API provided by framework

Consensus Policy

- Foundational App
    - Consortium guidelines for Consensus policy.
- Private App
    - Set by creator through API provided by framework

Foundation Code Abstraction Layer

Hyperledger Composer

- Business Modeling Language
- JavaScript wrapper for logic
- ACL list
- Loopback for API access generation
- Yeoman for Angular front end scaffolding

Node Ontogeny

- Guardian Nodes
    - Controlled by impartial agent of the consortium
    - Highly available and redundant across the major cloud providers (AWS, Azure/BlueMix)
- Veritas Nodes
    - Run by members of the consortium
- Peer Nodes
    - Run by non-consortium members
    - Primarily law firms and in house legal departments of large companies

Non-Node Access

Integra Gateways hosted by Veritas nodes

- Redundant RESTful endpoints.

High Performance

Public facing gateways must be fronted by Enterprise Service Bus (ESB)

- RabbitMQ

Containerization

- Docker
- Docker Composer

Resources

GitHub Repository

https://github.com/IntegraLedger

RocketChat Channel

Coming soon

ReadTheDocs Site

Coming soon

Jira Site

Coming soon

Hyperledger Website

http://hyperledger.org/

Hyperledger Fabric Website

http://hyperledger.org/projects/fabric

Hyperledger Composer Website

https://www.hyperledger.org/projects/composer

IBM BlueMix Blockchain as a Service

https://console.bluemix.net/docs/services/blockchain/index.html#gettingstartedtemplate

Roadmap

Draft pending input from external technologists and consortium members.

Version 0.5

- Update Architecture with feedback from external developers and consortium members
- Develop training materials
- Setup RocketChat Channel
- Setup Jira site
- Implement IBM Integra Node on IBM BlueMix
- Create Detailed Requirements
- Create detailed functional Spec
- Beta Release LMATId foundational App
- Alpha Release LMAT Document Registry POC App
-

Version 1.0

- Release LMATId foundational App
- Beta Release LMAT Document Registry POC App
- Implement 1.0 Framework Release
- Implement Hyperledger Indy support
- Develop Docker containers and deployment scripts
- Implement the first set of consortia POCs
- Implement IBM Integra Node on Amazon AWS
- Implement IBM Integra Node on Microsoft Azure
- Conduct largescale Security Audit

Version 1.x

- Add support for Direct SDK access to the Integra Ledger
- Obtain ISO certification
- Create integrations with Major 3rd party legal system
- Perform Largescale load and performance testing

References

1. *http://hyperledger.org/*
2. *http://hyperledger.org/projects/fabric*
3. *https://www.hyperledger.org/projects/composer*
4. *https://sovrin.org/*
5. *https://console.bluemix.net/docs/services/blockchain/index.html#gettingstartedtemplate*
6. *Hyperledger Fabric 1.0: Market Significance and Technical Overview* Available From: https://www.meetup.com/Hyperledger-Sydney/events/242004159/
7. *Blockchain development fundamentals on Hyperledger fabric - Matt Lucas (IBM*) Available From: https://www.safaribooksonline.com/library/view/oscon-2017-/9781491976227/video309376.html
8. *THE BLOCKCHAIN: A Guide for Legal and Business Professionals* Josais N. Dewey/Shawn S. Amuial/Jeffrey R. Seul 2016
9. *Mastering Blockchain* Imran Bashir 2017
10. *Data Modeling and Blockchain* Steve Hoberman 2017
11. *Kiss Principle* Available From: https://en.wikipedia.org/wiki/KISS_principle
12. *Blockchain for Dummies* Tiana Laurence 2017