SUPPORT AREAS

The Factoid Authority

Prepared for
Factom Guides

Prepared by
The Factoid Authority

THE FACTOID AUTHORITY

Contents

Support Areas 2

Factom Block-Explorer 2

FactomDB 2

Node Bootstrap Service 4

C# .NET Factom Client Library 4

Factom Light Node 5

Factomize! 5

Factom API Assist 5

Leveraging FactomDB to create a Factom node with superpowers! 5

FactomGPS 6

Factomd gRPC or Web Socket 6

Support Areas

We have identified the following projects to be implemented succeeding M3. Funding will be sought through a mixture of grants and the revenue generated from Authority Sets. We will also continue to support the Factom Community Test-net.

Please note this list is not exhaustive, due to the possibility of collaborations with other entities we have not disclosed everything.

Factom Block-Explorer

Blockchain explorers are crucial for any successful blockchain protocol, as it allows for an additional view into the blockchain for node operators.

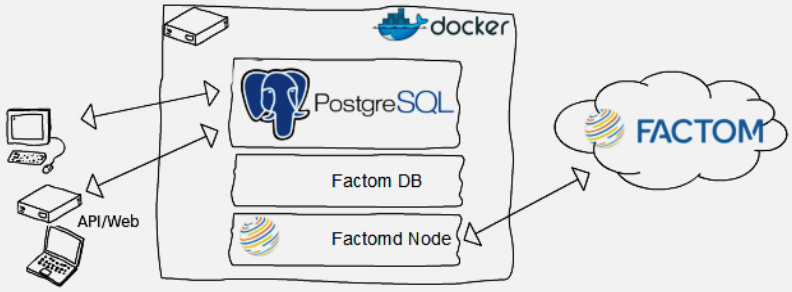
It is used for development and information relay across the web such as transaction confirmation, block height and more. Data is what creates a blockchain, but it is not the only thing of interest. The current Factom explorer covers the data aspect; TFA will look to expand beyond this.

In addition to the features of the current explorer, TFA's explorer will contain:

- Blockchain statistics such as those found here: **https://blockchain.info/charts**
- Identity Information
  - Ability for node operators to verify successful identity creation before campaigning
  - Visibility of all registered identities (Authority Set & others)
- Authority Information
  - Keeping records of historical Authority Set performance
- Realtime consensus
  - Monitoring Authority Set health from various data points to prove performance compliance
  - Ability for Authority Set operators to track their performance
  - Extendable to offer alert services

FactomDB

FactomDB will make reading Factom data easily accessible using SQL by storing indexable data directly on a Factom Chain. No Factom experience will be necessary, eliminating a major hurdle for many developers.



Factom can store 1K bytes of data, for $0.001. That's a lot of data, for recording time stamped events and metadata, locked up forever, in a way that can't be manipulated later. And for such a small fee.
Let's take a GPS tracker for example, fitted to a refrigerated truck. You need to know where the truck has been, when, and prove the contents were chilled at all times. If the goods being carried became a subject of investigation, insurance or otherwise, now it's possible to provide proof.

A GPS Tracker may send a message like this as represented in JSON:

```
{
        "id": 12345,
        "timestamp": 123456,
        gps: {
                Lat: 51.5074,
                Lon: 0.1278,
                Alt: 130,
                Speed: 36,
                Direction: 312,
        },
        Temp: -2.4,
        Event: "Movement"
}
```

That's about 120 bytes in compressed JSON. About 8 messages within a single 1K Factom entry. A more efficient approach, would be to use binary data. GPS Trackers typically do this to keep data transmission low but using incompatible formats. You might get 25 messages for every 1K byte.

Enter protocol-buffers...

Protocol Buffers allow you to express complex data structures into small compressed binary blobs with minimal overhead. Furthermore, new fields and data structures can be added without breaking compatibility.

For more details and examples, see **https://developers.google.com/protocol-buffers/**

If proof is required for an insurance claim e.g. - where a vehicles' refrigerated temperature fails to meet the required standard, or identifying which vehicles passed through a geographical location at a known time - a Relational Database, like the popular open source PostgreSQL is required and the fields of interest indexed. Factom DB will take care of this automatically. The user needs only to interface using SQL and no understanding of Factom is necessary.

PostgreSQL	FactomDB

Create table set	FactomDB will automatically create a new Factom chain and upload a protocol-buffers data structure as the first record based on all fields or tagged to be Factomized
Add a record	FactomDB will upload the new record to a Factom chain, serialized as protocol-buffers
Add a Factomized field/index	FactomDB will upload an updated protocol-buffers data structure as a special data entry
Delete or Edit Factomized field	Error!

Node Bootstrap Service

Any peer to peer service has a problem with being "bootstrapped"; in other words, finding the initial set of peers. Currently Factom operates by providing a set of "trusted" nodes in the Factomd configuration, but these nodes will be saturated in the future due to a hardcoded peer limit of 150 nodes.

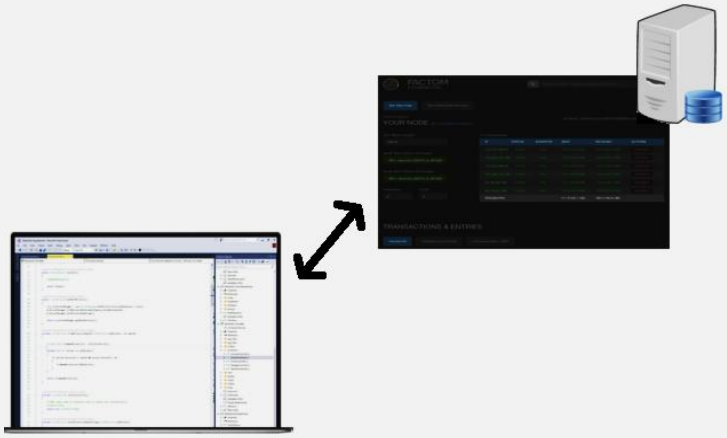
We will investigate designing a dynamic bootstrap service that benefits the network topology, by including 2 servers within 2 hops of an authority node, 2 servers of random peers who previously used the bootstrap service and 2 servers from the peer discovery process.
Note: Peer discovery process still yet to be defined.

C# .NET Factom Client Library

Factom is a blockchain solution for data. It's reference client, written in Golang, uses an RPC API provided by a Factom node (Factomd).

.NET is used a lot in business and C# is consistently in the top 5 language jobs in demand with several legacy business apps using it.

Factom needs a first class, C# client library; A DLL that can be plugged into any .NET app, run the demos to discover what new opportunities Factom can bring.



Requirements:

- DLL for all popular versions of .NET
- Full asynchronous stack
- LINQ Support
- Cursor support
- Cross platform
- 100% C# Managed code
- Automated build system, that will create the DLL of choice (docker image)
- NuGet integration

Factom Light Node

Nodes for smartphones and IoT.

The Factom Light Node Framework will be another area where the TFA will focus development efforts. Light nodes are crucial for developing lightweight applications to allow users to rapidly retrieve information on the Factom blockchain pertinent to only their specific application. We hope to help bridge the gaps to enterprise adoption in this area, as we envisage IoT to account for a considerable amount of the world's data consumption in the following years.

Factomize!

We will make efforts towards Factomizing large datasets within Star Catalogues for easy verification and retrieval. Currently, there are terabytes of useful data stored in disparate places that Factom can pull together. Some examples of such catalogues are Tycho, Tycho-2, and HIPPARCOS. We believe we can showcase the power and utility of Factom to governments and the scientific community and set a standard for organizing access to multi-spectral catalogs and alike.

Factom API Assist

Leveraging FactomDB to create a Factom node with superpowers!

FactomGPS

A C# library with optional API to provide GPS functions. Create a mapping application to track assets with proofs

Factomd gRPC or Web Socket

A real time data & console socket for Factomd

Steamr Integration

Streamr's IoT platform only supports Ethereum currently