Hyperledger Explorer Documentation

Explorer

Feb 02, 2021

Contents

HYPERLEDGER EXPLORER

CHAPTER 1

Overview of Hyperledger Explorer

CHAPTER 2

Table of contents

2.1 Introduction

Hyperledger Explorer is a tool for visualizing blockchain operations of the Hyperledger Fabric platform. It is the first ever blockchain explorer for permissioned ledgers, allowing anyone to explore the distributed ledger projects being created by Hyperledger's members from the inside, without compromising their privacy. The project was contributed by DTCC, Intel, and IBM. The ability to visualize data is of critical importance, in order to extract business value from it. Hyperledger Explorer provides this much needed functionality.

2.1.1 What are the key features of Hyperledger Explorer?

- Web application with a rich user interface, developed using latest technologies, such as ReactJS, Google material ui, web-sockets, and others.
- Blockchain module that can listen, and query a Hyperledger Fabric network.
- Utility that can:
  - Get the latest status blocks, network, and chaincodes, view blocks, and transactions.
  - Blocks and transactions metrics by hours, and minutes.
  - Search, and filter blocks, transactions by date range and channels.
  - Dynamically discover new channels and switch data presentation by channels.
  - Get real time notification of new blocks.

2.2 Hyperledger Explorer screens

2.2.1 Login screen

When user navigates to the home page of Hyperledger Explorer or enters default url **http(s)://<host>:<port>**, initially the login screen will be displayed to select the *Network Name*, and enter the user, and password based on the configuration in order to access the dashboard page.

Important: Expiration time of the JWT token, can be configured and by default is set to two hours.

2.2.2 Dashboard

Dashboard is the home page of the Hyperledger Explorer, and displays a set of panels that will show the latest activity of the Hyperledger Fabric network the explorer is configured to. We can see the navigation tabs: *DASHBOARD*, *NETWORK*, *BLOCKS*, *TRANSACTIONS*, *CHAINCODES*, *CHANNELS*. Along with the navigation tabs there are: *channel drop down*, *dark theme switch*, and *latest notification* icons. List of the peers can be seen, metrics, followed by latest activity, and transactions by organization panels. Each panel reflects the historical, metrics, and the latest activity per selected channel.

Note: To see another channel latest activity, select another channel from the drop down list.

2.2.3 Latest blocks notification panel

The latest block notification panel displays the block number, channel name, datahash, and the number of transactions per block. There is an *icon link* to show the latest block details, and is located in the right most corner of the panel. In the picture below you can see the block details accessed from the provided link.

2.2.4 Dark theme screen

The dark theme mode is another view of the Hyperledger Explorer, there can be some potential benefits by switching to it.

2.2.5 Network

Network screen consist of the list of the properties that channel is configured to.

2.2.6 Block list

Block list displays a list of the block attributes, and links to block, and transaction pop up details window. Please notice when mouse over it will display the actual block hash. Displayed in a table all the headers are sortable either descending or ascending depending of the direction selected. A filter can be applied to search by date range, and selected organization from the list.

2.2.7 Block details window

This screen represents in detail a block. You can see the creation of the block, number of transactions in this block, block hash, data hash, and the previous hash that connects to the previous block. For usability you can copy any of the hashes using the clipboard icon.

2.2.8 Filter result

By default block list will return one day woth of data, but you can modify the search criteria to search historical data, and apply also filter to the returned result. List can be sorted by one of desired direction, ascending or descending. To clear the global filter you can use the *Clear Filter* button. The *Reset* button can be used to reset the date range, and the organizations selection.

2.2.9 Transaction list

Transaction list screen has almost identical functionality as the block list screen, the number of rows per page, up to 100, can be selected from the drop down, next and previous buttons allow to navigate the direction of results, back or forward.

2.2.10 Transaction details

Transaction details is similar to block details, JSON data can be folded/unfolded for the preview purposes.

2.2.11 Chaincode list

Chaincode list displays the chaincode properties and has filter, and sort functionalities. Displayed list is for the current selected channel.

2.2.12 Channel list

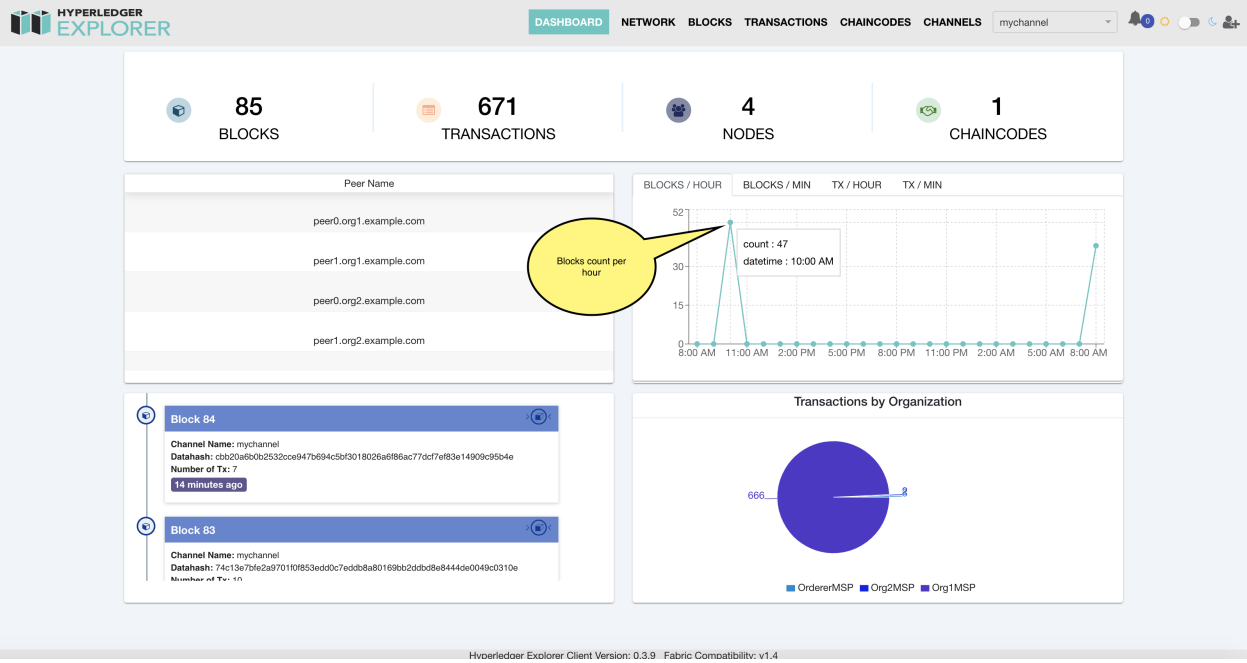
Channel list displays all the channels, and it's properties, has similar filter, and sort functionalities as chaincode list.

2.2.13 Metrics

Metrics panel has four tabs that will show the latest statistics of blocks, and transactions per selected channel. By hovering mouse over at any point of intersection in the graph will show the counts per hour, or minute. Click on any tabs below to see the metrics per block/transaction, hours or minute.

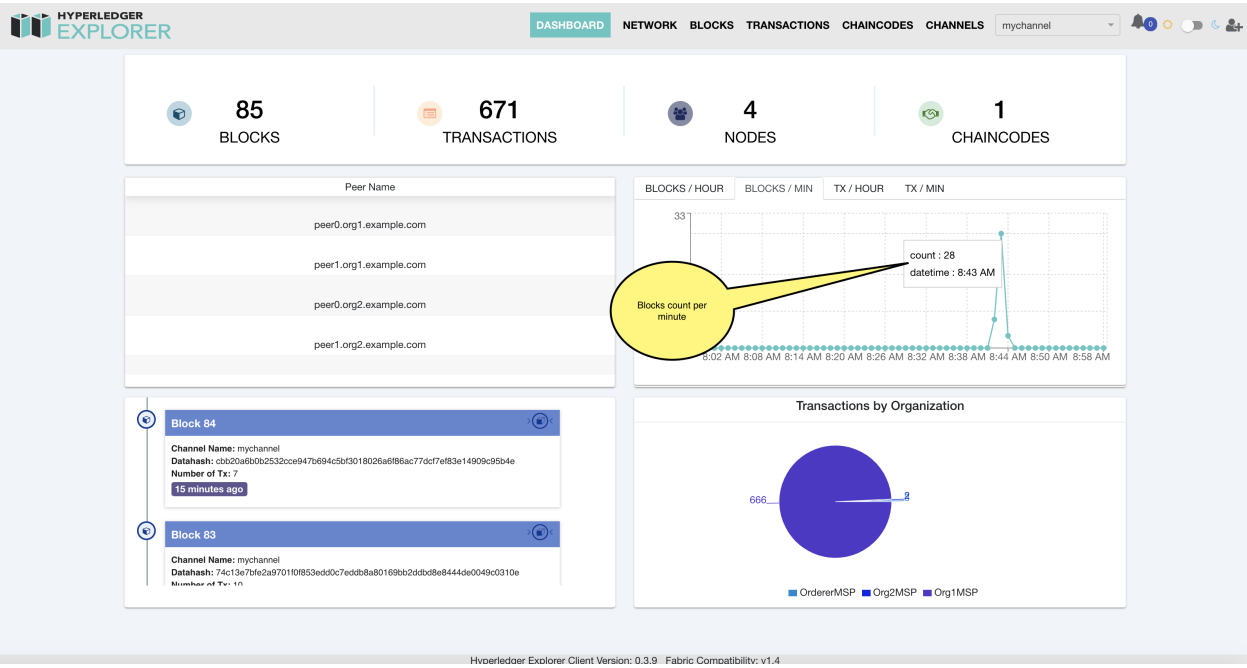
Blocks per hour

Displays the number of blocks added to fabric network in that period.



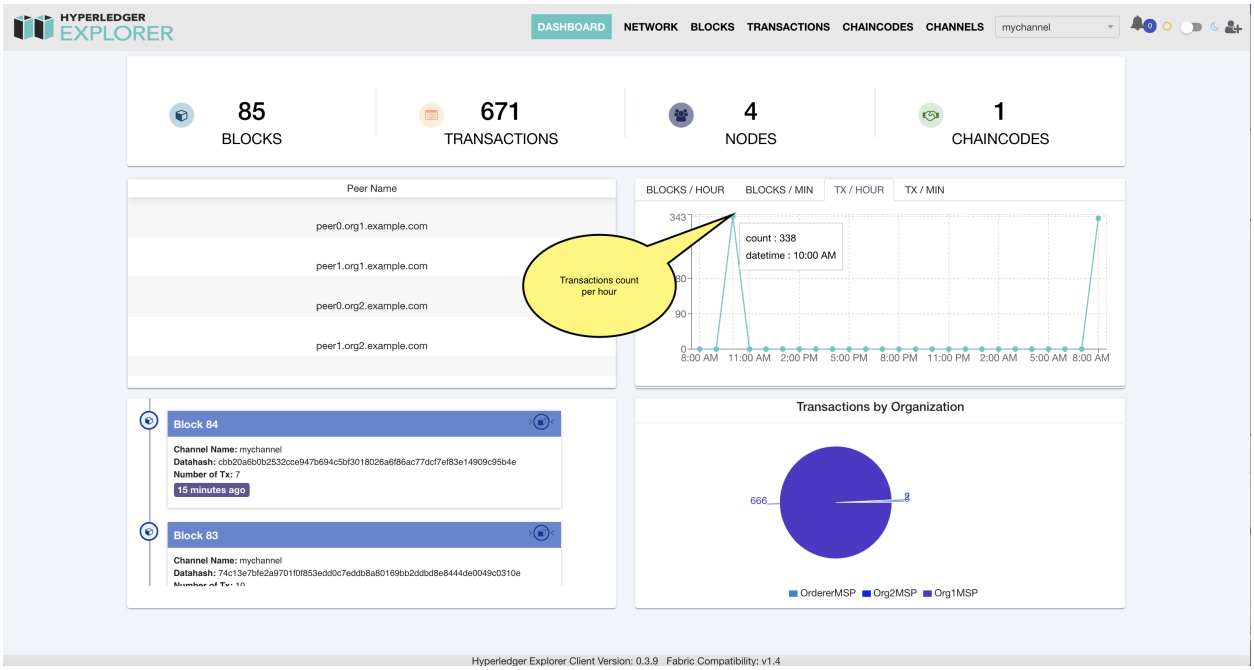
Blocks per minute

The number of transactions added to fabric network in that period.



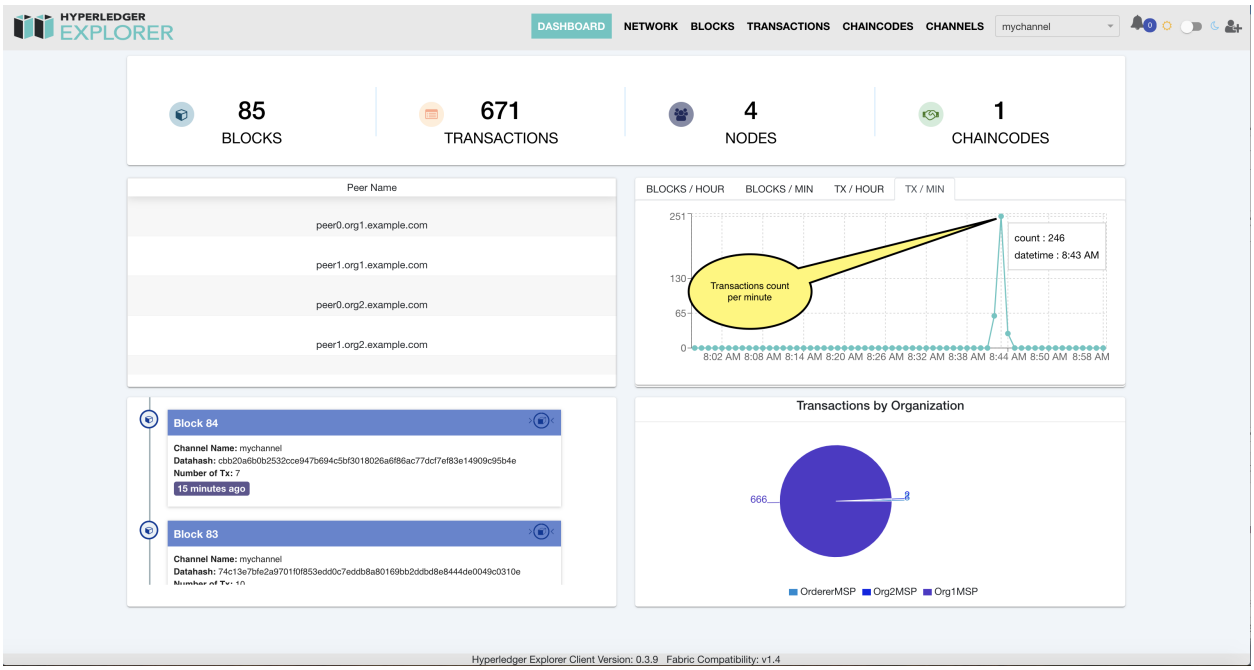
Transactions per hour

Displays the number of transactions added to fabric network in that period.



Transactions per minute

The number of transactions added to fabric network in that period.



2.3 Architecture Reference

2.3.1 Hyperledger Explorer Architecture

The high-level architecture of Hyperledger Explorer

2.3.2 Presentation Layer / Front End

When choosing a framework, or a library in most of the cases we look at the performance, maturity, and support this is where ReactJs was a benefit to us and used in our HLExplorer front end.

Why ReactJs?

- **Plainness**
  - **Component-based approach**
    - Use of well-defined lifecycle
    - JavaScript makes React very simple to learn, build professional web.
    - Use of a special syntax called JSX that allows to mix HTML with JavaScript.
- **Data Binding**
  - One-way data binding, control of the flow of data to components through the dispatcher, a one control point.
- **Performance**
  - React uses several clever techniques to minimize the number of costly DOM operations required to update the UI.
- **Testing**
  - Easy to test, rich set of libraries, like Jest, and React Testing Library
- **Straightforward to learn**
  - Easily understand React knowing JavaScript, and the good part: for react you just need basic knowledge of HTML and CSS.

The high-level architecture of presentation layer

2.3.3 Back End Layer

Main components of the HLExplorer back end can be listed as:

Webserver

Node.js is the backend framework for implementing the server-side components, and Express, a Web framework for Node.js. application. The main entry point of the Hyperledger Explorer is the Broadcaster class, that will initialize the application, WebSockets, create an Express server, and other processes to start the application.

Broadcaster class diagram shown in the image below.

REST API

The Hyperledger Explorer REST API provides endpoints to retrieve information, and metrics about the Hyperledger Fabric blockchain, such as blocks, transactions, nodes, chaincodes, channels, peers, and other information.

The REST API is designed to be used in the Node.js JavaScript runtime, and uses built-in middleware function in Express to parse incoming requests with JSON payloads and is based on body-parser.

Security on the Hyperledger Explorer REST API is enforced with a token. All requests made to the REST API must have a token that is generated by the API.

Overview

The REST API list of features include:

- **Authorization routes**
  - Login
  - Network list
- **Platform routes**
  - Transactions by organization(s)
  - Channel info
  - Channel list
  - Current channel
  - Change channel
  - Block by number
  - Peer status
- **Database routes**
  - Blocks counts
  - Block list
  - Transactions counts
  - Transaction list
  - Transaction information
  - Block and transaction list
  - Chaincode list
  - Peer list
  - Transactions per minute
  - Transactions per hour(s)
  - Blocks per minute
  - Blocks per hour(s)

Routes

There are three main routes defined, authorization, database, and platform.

Swagger

To learn more about REST API, and use an interactive API console to submit requests you can take advantage of the SWAGGER provided by the Hyperledger Explorer.

Hyperledger Explorer SWAGGER home page.

Gateway to Hyperledger Fabric

Due to the fact that Hyperledger Fabric network channel can constantly change, the peer, orderer and CA components, contributed by the different organizations in the network, will come and go, the need of reconfiguration for the Hyperledger Explorer becomes very difficult to maintain and get real time updates of the fabric network.

After the introduction of the "fabric-network" module by the Hyperledger Fabric SDK for NodeJS. Hyperledger Explorer uses Gateway, and a connection profile to connect to the fabric network. Read more about Gateway.

FabricGateway shown in the diagram below.

Configuration using Gateway and connection profile

One of the requirements to connect to a Hyperledger Fabric network to receive blocks, transactions, and other is to configure Hyperledger Explorer to be able to connect to the default fabric network. For this we tried to have a minimal as possible configuration by taking advantage of the latest Hyperledger Fabric Service Discovery, and Gateway. As of now Hyperledger Explorer is able to connect only to a single Hyperledger Fabric network, and we believe that in the next releases we may be able to have more than one network connected.

In previous versions we had a single file config.json file that was used to configure Hyperledger Explorer, after the minimal configuration we divided in two parts, config.json, and connection-profile, that described the network properties.

Sample Hyperledger Explorer configuration with one of the fabric sample network first-network, described below:

On another terminal:

```
cd blockchain-explorer/app/platform/fabric
```

Modify config.json to point to your first-network network connection-profile:

```
{
        "network-configs": {
                "first-network": {
                        "name": "first-network",
                        "profile": "./connection-profile/first-network.json"
                }
        },
        "license": "Apache-2.0"
}
```

```
"first-network" is the name of your connection profile, and can be changed to any␣
↪name.

"name" is a name you want to give to your fabric network, you can change only value␣
↪of the key "name".

"profile" is the location of your connection profile, you can change only value of␣
↪the key "profile"
```

(continues on next page)

(continued from previous page)

```
Modify connection profile in the JSON file first-network.json:
Change "fabric-path" to your fabric network disk path in the first-network.json file:
/blockchain-explorer/app/platform/fabric/connection-profile/first-network.json
Provide the full disk path to the adminPrivateKey config option, it usually ends with
↪"_sk", for example:

"/fabric-path/fabric-samples/first-network/crypto-config/peerOrganizations/org1.
↪example.com/users/Admin@org1.example.com/msp/keystore/
↪aaacd899a6362a5c8cc1e6f86d13bfccc777375365bbda9c710bb7119993d71c_sk"

"adminUser" is the the admin user of the network, in this case it's fabric CA or an␣
↪identity user.

"adminPassword" is the password for the admin user.

"enableAuthentication" is a flag to enable authentication using a login page, setting␣
↪to false
will skip authentication.
```

Database Services

This module is a set of API's that handle the interactions with the database. The layer is isolated from the HTTP request and response. Hyperledger Explorer at this time persisting data from the Hyperledger Fabric only, no delete operations involved.

The following API's are part of the database services:

1. PersistenceFactory, responsible for platform based blockchain.
2. Persist, has a set of getters, and setters to get services such as CRUDService, MetricServices, and PgService.
3. PgService is responsible to connect, and perform any data manipulation.
4. CRUDService, has a set of platform specific, and interacts with models that are storing data related to Hyperledger Fabric network blockchain.
5. MetricService, is in charge of computing statistics of the blockchain activity, per hours, minutes, and organizations.

Below diagram shows above mentioned API's.

Synchronizer

Incremental changes can have a big impact on application, and network performance, for this Hyperledger Explorer persists Hyperledger Fabric data in order to be able to explore historical, latest blocks, and see metrics per channels.

Synchronization process can be run in two modes:

1. Standalone sync, explorer UI unavailable.
2. Local (Run with Explorer)

The image below represents the synchronization process.

How synchronization process works?

Currently Hyperledger Explorer relies on Service Discovery to discover the network topology, and to issue queries to get:

- the MSPConfig of all organizations in the channel along with the orderer endpoints of the channel.
- the peers that have joined the channel.
- endorsement descriptor for given chaincode(s) in a channel.
- the local membership information of the peer that responds to the query.

Note: To enable Service Discovery the environment variables CORE_PEER_GOSSIP_BOOTSTRAP, and `CORE_PEER_GOSSIP_EXTERNALENDPOINT` needs to be added for each peer in the fabric network docker-compose.yaml file.

Once the sync process is started either in standalone, or local mode, Hyperledger Explorer will read the config.json, and will attempt to create a connection that was read from one of the connection profiles, for example first-network.json.

The first step after establishing the connection, explorer will get the list of the channels, and then loop each channel to get the list of the peers, organizations, chaincodes for a specific channel that will be passed to the Hyperledger Fabric Client SDK.

Each channel will be synchronized with explorer database, for the following:

- existence of the channel
- blocks per channel
- transactions
- chaincodes per channel
- peers/orderers

Configuration

- Modify explorerconfig.json to update sync properties:
  - sync type (`local` or `host`), platform, blocksSyncTime(in min) details.
- Sync Process Configuration
  - Ensure same configuration in Explorer explorerconfig.json if sync process is running from different locations.

Host (Standalone)

```
{
        "sync": {
        "type": "host"
        }
}
```

Local (Run with Explorer)

```
{
        "sync": {
        "type": "local"
        }
}
```

- Running Sync Process

Run host (Standalone)

- From new terminal (if Sync Process in Standalone).
  - `cd blockchain-explorer/`
  - `./syncstart.sh` (it will have the sync node up).
  - `./syncstop.sh` (it will stop the sync node).

Run local (Run with Explorer)

- From a new terminal:
  - cd blockchain-explorer/
  - `./start.sh` (it will have the backend up).
  - `./start.sh debug` (it will have the backend in debug mode).
  - `./start.sh print` (it will print help).
  - Launch the URL `http(s)://<host>:<port>` on a browser.
  - `./stop.sh` (it will stop the node server).

Attention:

- Please restart Explorer if any changes made to explorerconfig.json.
  - If the Hyperledger Explorer was used previously in your browser be sure to clear the cache before relaunching.
  - If Hyperledger Fabric network is deployed on other machine, please toggle `DISCOVERY_AS_LOCALHOST` in `start.sh / syncstart.sh to 'false'`.

WebSockets

Websocket API is used to push information from the server to the clients. Information about new blocks, stats. are pushed via Websocket API. This is a convenient API that reduces load on clients and server.

The image below illustrates how the notification is displayed when new blocks are received from the blockchain network, and a link to the new block details is provided in the notification panel.

Logs

Logging is an important part of supporting Hyperledger Explorer in the complete application life cycle. Logs are created to debug, and provide information while troubleshooting, and detect problems, and any failure of the application.

Why do we log?

- Performance, we care about performance, and to measure it we count on logging.

- Debugging, we need a mechanism to see when, and where the error occurred, and under what conditions.
- Error tracking, when errors do occur, we need to know when they started, and how often they occurred.
- Analyzing, logs are valuable sources of information. We can analyze logs to discover usage patterns and make decisions.
  - **There are two directories created by the application, one for standalone sync, and when running with explorer**
    * `$blockchain-explorer/logs/.`
    * `$blockchain-explorer/logs/sync.`
  - **Both have same subdirectories**
    * `app/.`
    * `db/.`
    * `console/.`

Application logs

Information, debug, error and other types of events, these logs are located in `$blockchain-explorer/logs/app/` directory.

Database logs

Useful information, debug, and errors are recorded during the CRUD operations, logs are located in `$blockchain-explorer/logs/db` directory.

Console logs

Different levels of messages to stdout and stderr, all the console logs are also redirected to a console.log files for the auditing purposes, location the the logs are in `$blockchain-explorer/logs/console` directory.

Note: Logs are rotated every 7 days.

2.3.4 Database Layer

Hyperledger Explorer uses PostgreSQL database. The information about blocks, transactions, channels etc will be stored in this database. This is a mature database for real-time web applications as the updates will be retrieved from database instead of the application polling data from the Hyperledger Fabric network.

The following diagram shows a high level view of the Hyperledger Explorer data model.

Note: Please note, the connecting lines are just for info purposes to illustrate the relation, there are no constraints defined in the current database.

Physical schema

The script explorerpg.sql describes Hyperledger Explorer database. Creation of the database is a mandatory step and it is done by running the script createdb.sh, detailed steps and instructions are provided in the README.md file.

2.4 Getting Started with Hyperledger Explorer

Hyperledger Explorer can run in your local environment or using docker. Minimum configuration allows you to get it setup up, and running in short time. There are well defined steps and instructions on how to Get Started with Hyperledger Explorer how to get you setup and run HL Explorer successfully. Along with the instructions you may find useful troubleshooting notes, that were collected, and or submitted by community through the mail listings or Rocket Chat hyperledger-explorer channel.

Important: See Release Notes on supported Hyperledger Fabric and Explorer versions.

2.4.1 Running Hyperledger Explorer in kubernetes

Note: Hyperledger Explorer project team did not test, or setup to run explorer in kubernetes, a contributor did a tremendous job in setting it, and added an explanation on how to set it up. If you're interested to experiment, and willing to give a try, please see running explorer in kubernetes.

2.5 Repositories/Links

2.5.1 Hyperledger Explorer Repositories

- GitHub:
  - Hyperledger Explorer github repository.
- Docker:
  - Hyperledger Explorer docker repository.
  - Hyperledger Explorer PostgreSQL docker repository.

2.6 Contributing to Hyperledger Explorer

We welcome contributions to the Hyperledger Explorer Project in many forms like bug reports, feature requests, documentation updates, code.

2.6.1 Getting a Linux Foundation account

In order to participate in the development of the Hyperledger Explorer project, you will need a Linux Foundation account.

You will need to use your LF ID to access to all the Hyperledger community tools, including:

- Jira.
- RocketChat.
- Mailing list.

2.6.2 Bug Reports and Feature Requests

We track all bugs and feature requests on Hyperledger Explorer Jira. Please send a detailed bug report with relevant logs and steps to reproduce the issue when you encounter an issue. We always appreciate such bug reports as it reduces our effort and will help us fix the bug easily. Please use the search functionality to ensure that the bug/feature request you are trying to file does not exist already. If it exists, you can always add additional information to the JIRA issue in the comments sections and start watching for updates. All the issue here will be added to our backlog and prioritized in the upcoming sprints. You can contact the developers for any general problems or questions on Hyperledger Explorer Chat.

2.6.3 Communications

We use RocketChat for communication and Zoom meeting™ for screen sharing between developers. Our development planning and prioritization is done in JIRA, and we take longer running discussions/decisions to the mailing list.

2.6.4 Contribution guide

Prerequisites

Following are the software dependencies required to install and run Hyperledger Explorer:

- Nodejs 8.11.x (Note that v9.x is not yet supported)
- PostgreSQL 9.5 or greater
- jq (command-line JSON processor)
- Linux-based operating system, such as Ubuntu
- MacOS

Verified Docker versions supported:

- Docker CE 18.09.2 or later
- Docker Compose 1.14.0

Working on fixing issues and working stories

Review the issues list and find something that interests you. Start with something relatively straight forward and achievable, and that no one is already assigned. If no one is assigned, then assign the issue to yourself. Please be considerate and rescind the assignment if you cannot finish in a reasonable time, or add a comment saying that you are still actively working the issue if you need a little more time.

Submitting your fix

If you just submitted a JIRA for a bug you've discovered, and would like to provide a fix, we would welcome that gladly! Please assign the JIRA issue to yourself, then you can submit a change request (PR), please follow guidance provided by CONTRIBUTING.md.

Copyright Notices

There should be a single LICENSE file in the top-level directory that contains the full text of the Apache License here.

Note: In the individual files, please use the following line:

```
/*
   SPDX-License-Identifier: Apache-2.0
*/
```

2.6.5 Related Topics

Maintainers

Name	GitHub	RocketChat	email
Adam Kwan	adamk1230	adamk1230	adamk1230@gmail.com
Atsushi Neki	nekia	nekia	atsushin@fast.au.fujitsu.com
David Huffman			dshuffma@us.ibm.com
Jeeva Sankarapandian	jeevasang	jeevas	jsankarapandian@dtcc.com
Mekia Edwards	edwardsm26	edwardsm26	dev.edwardsm@gmail.com
Nik Frunza	nfrunza	nfrunza	nfrunza@gmail.com
Satheesh Kathamuthu	xspeedcruiser	satheeshk	satheesh.ceg@gmail.com
Umapathi Madiraju	umadiraju	umadiraj	umapathi.madiraju@gmail.com
Vinita Chinoy	vchinoy-da	vchinoy	vinitachinoy@yahoo.com

Setting up the development environment

Overview

Hyperledger Explorer has been developed in Ubuntu, and macOS environments

Please follow the instructions for Ubuntu builds, below.

Prerequisites

- Git client
- Go - version 1.11.x
- (macOS) Xcode must be installed
- Docker - 17.06.2-ce or later
- Docker Compose - 1.14.0 or later
- Pip
- (macOS) you may need to install gnutar, as macOS comes with bsdtar as the default, but the build uses some gnutar flags. You can use Homebrew to install it as follows:

```
brew install gnu-tar --with-default-names
```

```
pip install --upgrade pip
```

Steps

Set your GOPATH

Make sure you have properly setup your Host's GOPATH environment variable. This allows for both building within the Host and the VM.

In case you installed Go into a different location from the standard one your Go distribution assumes, make sure that you also set GOROOT environment variable.

Cloning the Hyperledger Explorer source

Contribution Workflow

The documentation is maintained using a "*contributor workflow*" where everyone without exception contributes changes proposals using "*pull-requests*".

This facilitates social contribution, easy testing, and peer review.

To contribute changes, use the following workflow:

- Fork the repository.
- Clone your fork to your computer.
- Create a topic branch and name it appropriately. Starting the branch name with the issue number is a good practice and a reminder to fix only one issue in a Pull-Request (PR).
- Make your changes, adhering to the documentation conventions described below. In general a commit serves a single purpose and diffs should be easily comprehensible. For this reason do not mix any formatting fixes or typo fixes with actual documentation changes.
- Commit your changes using a clear commit message.
- Test your changes locally before pushing to ensure that what you are proposing is not breaking another part of the doc.
- Push your changes to your remote fork (usually labeled as *origin*).
- Create a pull-request (PR) on the Hyperledger Explorer doc repository. If the PR addresses an existing Jira issue,include the issue number in the PR title in square brackets (for example, *[BE-234]*).
- Add labels to identify the type of your PR. _For example, if your PR fixes a bug, add the "bug" label.
- If the PR address an existing Jira issue, comment in the Jira issue with the PR number.
- Ensure your changes are reviewed. Select the reviewers you would like to review your PR. If you don't know who to choose, simply select the reviewers proposed by GitHub or leave blank and let us know on Hyperledger Explorer chat.
- Make any required changes on your contribution from the reviewers feedback. Make the changes, commit to your branch, and push to your remote fork.
- When your PR is validated, all tests passed and your branch has no conflicts with the target branch, you can "squash and merge" your PR and you're done.

Pull Requests

The process described here has several goals:

- Maintain documentation quality
- Fix problems that are important to users
- Engage the community in working toward the best possible documentation
- Enable a sustainable system for maintainers to review contributions
- Further explanation on PR & commit messages can be found in this post: How to Write a Git Commit Message.

Please follow these steps to have your contribution considered by the approvers:

- Ensure all commits have a Sign-off for DCO, as described in DCO.md.
- After you submit your pull request, verify that all status checks are passing.

While the prerequisites above must be satisfied prior to having your pull request reviewed, the reviewer(s) may ask you to complete additional writing, or other changes before your pull request can be ultimately accepted. All changes must be code reviewed. For non-approvers this is obvious, since you can't commit anyway. But even for approvers, we want all changes to get at least one review, preferably (for non-trivial changes obligatorily) from someone who knows the areas the change touches. For non-trivial changes we may want two reviewers. The primary reviewer will make this decision and nominate a second reviewer, if needed. Except for trivial changes, PRs should not be committed until relevant parties (e.g. owners of the subsystem affected by the PR) have had a reasonable chance to look at PR in their local business hours.

If an approver intends to be the primary reviewer of a PR they should set themselves as the assignee on GitHub and say so in a reply to the PR. Only the primary approver of a change should actually do the merge, except in rare cases (e.g. they are unavailable in a reasonable timeframe).

If a PR has gone 2 work days without an approver emerging, please ask on Hyperledger Explorer Rocketchat

Building Hyperledger Explorer

The following instructions assume that you have already set up your *development environment*.

To build Hyperledger Explorer:

```
cd blockchain-explorer
git checkout <branch-name>
./main.sh clean
./main.sh install
```

Note: Ask in chat what is the branch name of the latest being in development

Running the unit tests

Use the following sequence to run all unit tests

```
cd blockchain-explorer
./main.sh clean
./main.sh install
./main.sh test
```

Azure DevOps

Hyperledger Explorer uses Azure DevOps to automate the builds, run tests, and code coverage. After the pull request is created an automated job will be triggered defined by the Azure pipeline script.

Coding guidelines

Coding javascript

We code in javascript™ and follow the Airbnb JavaScript Style Guide, and Google JavaScript Style Guide.

PostgreSQL Coding Conventions

Keep PostgreSQL coding guidelines and follow the PostgreSQL The SQL Language Coding Conventions.