Plenny DLSP Module

Operating Manual

Setting up the open-source add-on module to connect your Lightning Node and Ethereum Wallet to the DLSP Module

Guide for Liquidity Makers to leverage the Dapp's Capacity Market and for Lightning Oracles to use the Decentralized Oracle Network

Plenny Community | plenny@unstoppable.email

plenny.link | plenny.crypto | dplenny.crypto

Contents

Part I – Technical Implementation

Part II – Operational Concept

Part I Technical Implementation

1 Compatibility

In general, a Bitcoin Core and a BOLT-compliant Lightning Node are required. BOLT stands for Basis of Lightning Technology. The hardware and software requirements for Lightning Nodes correspond to the standard specifications.

Currently, any Lightning Node compatible with the Lightning Network Daemon (LND) starting from lnd 0.11.1-beta can use the DLSP module V.3.1.3-Beta.

In addition, an Ethereum wallet (i.e. MetaMask) and an Ethereum Network endpoint that supports Layer 2 (L2), more specifically Arbitrum One, are required.

1.1 Integration

Liquidity Makers and Lightning Oracles download the same open-source software: the Plenny DLSP module.

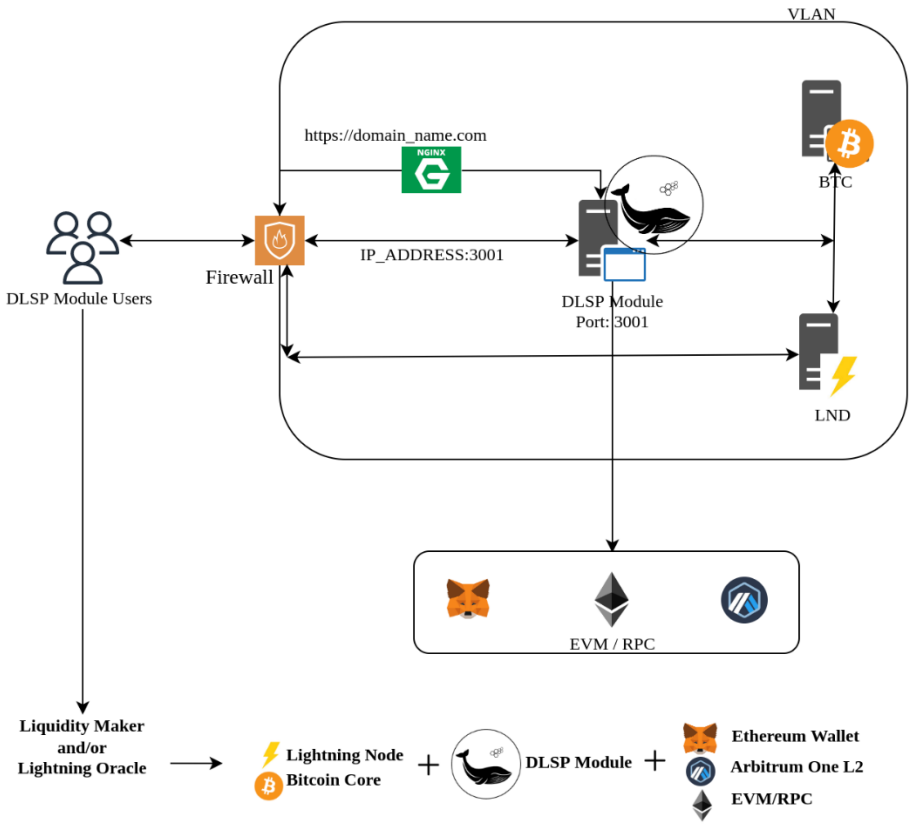
For integration, participants install an executable application on their computer. In the process, the Ethereum wallet's mnemonic phrase always remains secret and under the user's control.

2 Technical Prerequisite

In order to successfully install and run the DLSP module, some technical prerequisites must be met. The service will not be executed/proceed until all requirements are satisfied.

2.1 Lightning Node DLSP Module Diagram

The DLSP module is embedded into the crypto ecosystem using Web 3.0 as well as Web 2.0 technologies.



2.2 Bitcoin Node

Bitcoin Core Documentation:

https://bitcoin.org/en/full-node

https://github.com/bitcoin/bitcoin

BTCPay Server Documentation:

https://docs.btcpayserver.org/FAQ/LightningNetwork/

2.3 Lightning Network Daemon (LND)

The LND software component is an open-source implementation of a Lightning Network node by Lightning Labs Inc., Palo Alto, California, United States:

https://docs.lightning.engineering/lightning-network-tools/lnd

https://github.com/lightningnetwork/lnd

Lightning Node Setup:

https://wiki.ion.radar.tech/tutorials/nodes/

https://wiki.ion.radar.tech/tutorials/nodes/lighting-at-home

https://www.youtube.com/watch?v=rf-GvVYuWa8

Lightning Node Management:

https://www.lightningnode.info/

Lightning Wallets & Tools:

It is recommended to access your Lightning wallet directly from the command line ("lncli") instead of using Lightning browser extensions. The Lightning wallet allows you to manage your account balance in Satoshi (sat). In addition, there are several lightning tools for improving the user experience, but not all of them are compatible with LND.

https://www.lopp.net/lightning-information.html

2.4 Ethereum Wallet

To access the Dapp, go to dplenny.crypto and log in with your dedicated Ethereum wallet.

Use MetaMask: https://metamask.io/

Hard wallets are not yet supported.

2.5 Ethereum L2: Arbitrum Network

The DLSP module requires an Ethereum RPC endpoint that supports the Arbitrum Network on L2. There are three ways to connect to this optimistic rollup solution:

- Recommended: Use an Ethereum Remote Procedure Call (RPC) protocol that supports Arbitrum.

  RPC Alchemy: https://docs.alchemy.com/alchemy/

  RPC Infura: https://infura.io/docs

- Run your own Ethereum Node (EVM) and connect to an RPC-provider that supports Arbitrum:

  EVM: https://ethereum.org/en/developers/docs/evm/

  Make sure your EVM is started with option "--http" to enable JSON-RPC over HTTP protocol. A full node is recommended.

- Use your own Arbitrum Node (AVM):

  https://developer.offchainlabs.com/docs/avm_design

  https://developer.offchainlabs.com/docs/installation

  https://developer.offchainlabs.com/docs/running_node

2.6 SSL Certificate

The DLSP module requires an SSL to establish a secure connection to the Internet. It is recommended to use a CA-signed certificate from a third-party provider instead of a self-signed certificate. However, a self-signed certificate is equally sufficient.

Resources:

https://www.openssl.org/

https://letsencrypt.org/

3 Setup the DLSP Module

The Plenny Community provides binary files for installing the DLSP module on Windows and Linux operating systems.

Download the latest source release for Windows and Linux here:

https://github.com/PlennyPL2/Decentralized-Lightning-Service-Provider_DLSP/releases/

3.1 Operating Systems

Windows:

Execute the PlennyDLSP.exe file to set up and run the DLSP module.

Linux:

Before running the binary, you must export the .env environment variables. Use the script below:

```
export $(grep -v '^#' .env | xargs)
```

First way to execute the binary:

```
./PlennyDLSP
```

Second way to execute the binary:

Fill in the fields in the run.sh file and run the following command in your shell:

```
./run.sh
```

3.2 Installation Steps

- Download and unzip the source file
- Configure the .env file with your required parameters
- Upload your SSL certificates into server/certificates location
- Run “npm install”
- Execute the DLSP module binary

3.3 Environment File Configuration

The default environment variable file is placed in the directory. The following syntax rule applies to the .env file:

- VAR=VAL
- Line beginning with “#” are processed as comments and ignored
- Blank lines are ignored

The default .ENV file allows users to configure their own environment variables. It retrieves the information from the smart contracts to check the use case (i.e. Liquidity Maker or Lightning Oracle Validator) for connection to Plenny. Depending on the basic settings, the operating system, service level, and user type are detected. The settings are divided into different sections enabling users to configure the DLSP module step by step.

3.3.1 Configuration Port Number

To run the DLSP module on your server, provide the following settings:

```
PORT=<Your_Port_Numer>
```

Example:

```
PORT=3001
```

The default port number for the DLSP module is 3001. This port number is an optional variable.

If you want to run multiple instances of the DLSP module, change the port number and start the application one by one.

3.3.2 Bitcoin Node Configuration

The environment variables listed below are used to configure your Bitcoin node:

```
BTC_HOST=<Your_Bitcoin_Host_Address>
BTC_PORT=<Your_Bitcoin_node_RPCPort>
BTC_USERNAME=<Your_Bitcoin_node_Username>
BTC_PASSWORD=<Your_Bitcoin_node_Password>
BTC_VERSION=<Your_Bitcoin_daemon_Version>
BTC_NETWORK=<Your_Bitcoin_daemon_Network>
```

Example:

```
BTC_HOST=192.168.0.1
BTC_PORT=8332
BTC_USERNAME=bitcoinrpc
BTC_PASSWORD=mystrongpassword
BTC_VERSION=0.21.0
BTC_NETWORK=mainnet
```

3.3.3 Ethereum Configuration

The environment variables listed below are used to configure your Ethereum access point:

RPC Configuration:

```
LOCAL_RPC_URL= <Your_Ethereum_remote_RPC_endpoint>

SOCKET_RPC_URL=<Your_Ethereum_remote_RPC_WebSocket_endpoint>

L1_RPC_URL=< Your_Ethereum_remote_RPC_endpoint>

ETH_PRIV_KEY=<Your_Ethereum_Wallet_Private_Key_to_Address>
```

Example:

```
LOCAL_RPC_URL=https://arb-rinkeby.g.alchemy.com/v2/xxxx

SOCKET_RPC_URL=wss://arb-rinkeby.g.alchemy.com/v2/xxxx

L1_RPC_URL=https://rinkeby.g.alchemy.com/v2/xxxx

ETH_PRIV_KEY=4xxxx
```

Ethereum Wallet:

- The private key of the Ethereum wallet must match with the profile used on the Dapp.
- It is recommended to use a dedicated Ethereum wallet/address for all use cases related to Plenny.
- When an “incorrect tx nonce” error occurs, the DLSP module automatically synchronizes the transaction nonce and executes the transaction in a few seconds. Any pending transaction will be executed in the next channel iteration.

3.3.4 LND Configuration

The environment variables listed below are used to configure your node using the Lightning Network Daemon (LND):

```
LND_ADMIN_MACAROON_HEX=<HEX_Value_of_admin_macaroon>

LND_TLS_CERT_HEX=<HEX_Value_of_TLS_certificate>

LND_WALLET_PASS=<Your_LND_Wallet_Password>

LND_HOST_PORT=<Your_LND_RPC_Port_Number>
```

Example:

```
LND_ADMIN_MACAROON_HEX=02010a108c7f2646a692c3d592f3c31
b30a108c7f2646a692c3d592f3c31b3

LND_TLS_CERT_HEX=2d2d2d20a4d4949434954430a4d49494349544
354430a4d4949434544430a4d4949454430a4d4949434

LND_WALLET_PASS=mylndpassword

LND_HOST_PORT=192.168.0.1:10009
```

Find your HEX value from the macaroon using the following command:

```
xxd -p -c2000 admin.macaroon

xxd -p -c2000 tls.cert
```

3.3.5 Profit Configuration

The environment variable listed below is used to enable automatic closing of submarginal channels:

```
ALLOW_AUTO_CLOSING_CHANNELS=true/false
```

Example:

```
ALLOW_AUTO_CLOSING_CHANNELS=true
```

NB: The provision of inbound capacity may become unprofitable once the licensing period has expired. Thus, these channels become profitless in terms of Royalties and Channel Rewards paid in PL2.

However, profitless channels can still be profitable as the Liquidity Maker earns routing fees via the LN if the channel stays open longer.

3.3.6 Logging Configuration

For monitoring purposes and troubleshooting, the DLSP module uses Winston-defined log levels to print into the standard output (console). Each level defines the severity of a logged incident. To control what is being printed in the console, see the environment variable called LOG_LEVEL. It takes a string and defines what types of incidents are printed. For example, setting LOG_LEVEL=info will print any incident starting from info level and above.

The log level hierarchy defined by Winston is as follows:

```
const levels = {
  error: 0,
  warn: 1,
  info: 2,
  http: 3,
  verbose: 4,
  debug: 5,
  silly: 6
};
```

Notice the values, which signify the severity associated with the level, such that the lower the severity of the logged event, the higher the value. So, a debug message is considerably less important than a warning message.

In addition, the DLSP module transports any incident of type error into a separate file error.log inside the installation directory /server/logs/. This file contains any error incident that has occurred in the past.

```
LOG_LEVEL=const levels
```

Example:

```
LOG_LEVEL=info
```

Moreover, the DLSP module also provides an access.log via API. This list of all requests for individual endpoints is useful for collecting analytical data.

Resources:

https://www.npmjs.com/package//winston

https://github.com/winstonjs/winston

3.3.7 RPC Request Configuration

RPC providers limit the response rate. Depending on the account type, users encounter a rate limit response, with the request being repeated automatically after a short delay. This behavior can be configured by setting the following options:

MAX_RETRIES = Number of attempts the client will make to resend a rate-limited request before giving up.

RETRY_INTERVAL = Minimum time to wait between consecutive retries, in milliseconds.

RETRY_JITTER = Random amount of time is added to the retry delay to avoid additional rate errors caused by too many concurrent connections, with the number of milliseconds chosen between 0 and this value.

CHANNEL_FAILED_ATTEMPTS_LIMIT = Maximum number of failed attempts before a pending channel is ignored for further processing.

DELAY_CHANNEL_PROCESSING = Minimum time to wait for a channel processing to reduce requests burstiness, and prevent RPC rate limits, in milliseconds.

```
MAX_RETRIES=<Number_of_Retries>
RETRY_INTERVAL=<Interval_in_Milliseconds>
RETRY_JITTER=<Delay_in_Milliseconds>
CHANNEL_FAILED_ATTEMPTS_LIMIT=<Number_of_Failed_Attempts>
DELAY_CHANNEL_PROCESSING=<Delay_in_Milliseconds>
```

Example:

```
MAX_RETRIES=3
RETRY_INTERVAL=1000
RETRY_JITTER=250
CHANNEL_FAILED_ATTEMPTS_LIMIT=3
DELAY_CHANNEL_PROCESSING=250
```

3.3.8 Configuration Summary

Below is a sample ENV file which is used by the DLSP module:

```
# Port Number Configuration
PORT=Your_Port_Number

# Configuration for connecting to Bitcoin node
BTC_HOST=Your_Bitcoin_node_Host
BTC_PORT=Your_Bitcoin_node_Port
BTC_USERNAME=Your_Bitcoin_node_Username
BTC_PASSWORD=Your_Bitcoin_node_Password
BTC_VERSION=Your_BTC_Version
BTC_NETWORK=Your_used_Network

# Configuration for connecting to Ethereum blockchain
LOCAL_RPC_URL=https://Your_Ethereum_remote_RPC_endpoint
SOCKET_RPC_URL=wss://Your_Ethereum_remote_RPC_WebSocket_endpoint
L1_RPC_URL=Your_Ethereum_remote_RPC_endpoint
ETH_PRIV_KEY=Your_Ethereum_Private_Key_to_Address

# Configuration for connecting to LND node
LND_ADMIN_MACAROON_HEX=HEX_Value_of_admin_macaroon
LND_TLS_CERT_HEX=HEX_Value_of_TLS_certificate
LND_WALLET_PASS=Your_LND_Wallet_Password
LND_HOST_PORT=Your_LND_RPC_URL:Your_LND_RPC_Port_Number

# Profit Configuration
ALLOW_AUTO_CLOSING_CHANNELS=true/false

# DLSP Logging Configuration
LOG_LEVEL=const levels

# RPC Request Configuration
MAX_RETRIES=Number_of_Retries
RETRY_INTERVAL=Interval_in_Milliseconds
RETRY_JITTER=Delay_in_Milliseconds
CHANNEL_FAILED_ATTEMPTS_LIMIT=Number_of_Failed_Attempts
DELAY_CHANNEL_PROCESSING=Delay_in_Milliseconds
```

Setup is completed via the Dapp's UI by performing the initial verification. However, the system and network must be configured too.

3.4 System & Network Security

The DLSP module is reachable over HTTPS for secure communication.

3.4.1 Install Self-Signed Certificate

To generate a self-signed certificate, run the following commands in your shell.

#openssl genrsa -out key.pem

#openssl req -new -key key.pem -out csr.pem

#openssl x509 -req -days 9999 -in csr.pem -signkey key.pem -out cert.pem

Update the ".pem"-files in the server/certificates folder on the server where the DLSP module is hosted.

3.4.2 Secure Endpoint

Make sure your endpoint for the DLSP module is accessible through HTTPS. Non-secure http-connections are not supported. The SSL configurations listed below can be used:

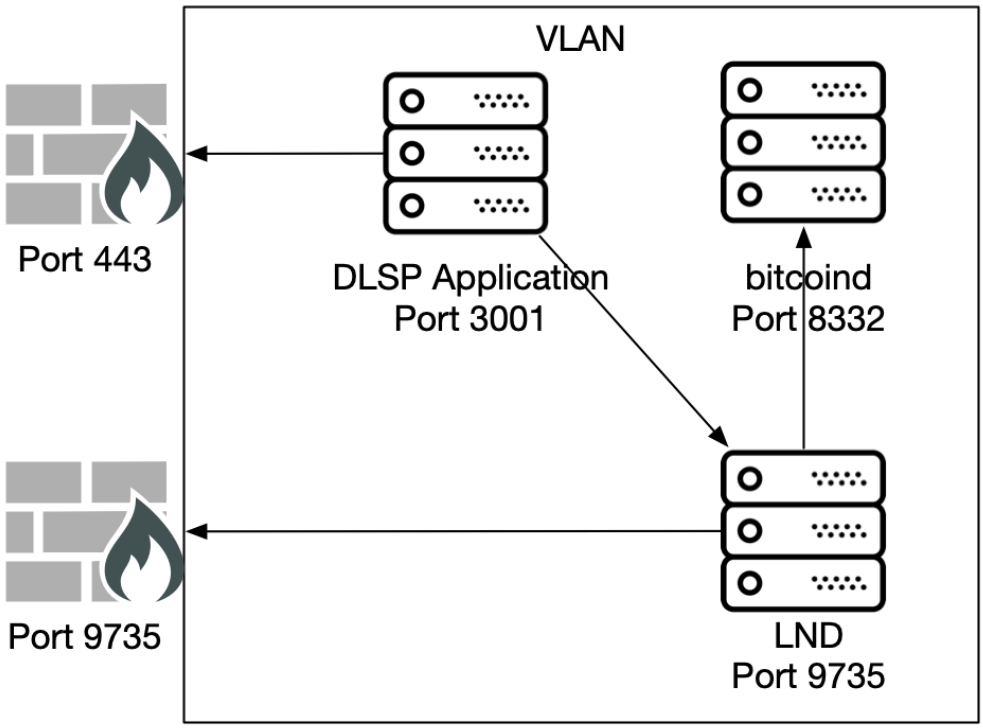
- Dockerised solution using docker-compose [nginx server with letsencrypt]
- Cloud load balancer with cloud provider's SSL [For cloud-hosted Lightning Nodes]
- Examples for using a proxy webserver:

  HAProxy: https://www.haproxy.org/

  Nginx: https://www.nginx.com/

3.4.3 Firewall Settings

To ensure high security, open HTTPS 443 and the LND port 9735 to the public internet.



4 Access the Dapp

4.1 Blockchain Domains

Plenny is assessable via blockchain domains on the decentralized web.

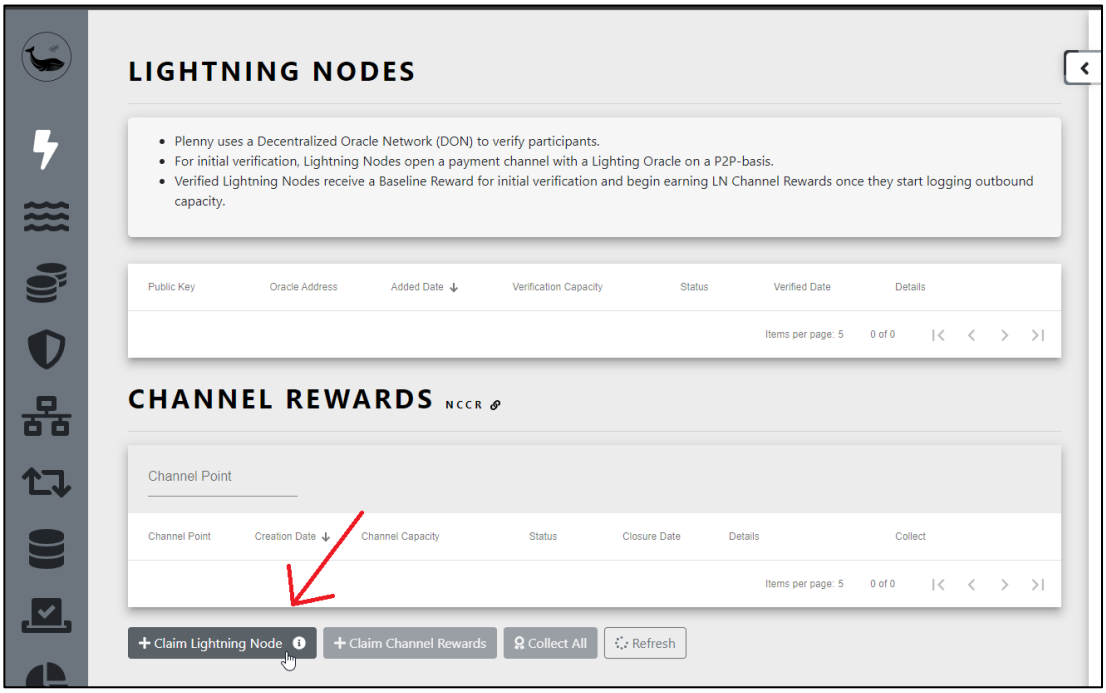
- Plenny HP Web 2.0: Plenny.link is the traditional homepage and runs on Web 2.0.
- Plenny HP Web 3.0: Plenny.crypto is Plenny's decentralized homepage, running on a blockchain domain and providing project-related content via decentralized storage using IPFS. Go to Plenny.link to find out how to access blockchain domains and decentralized websites.
- Plenny Dapp Web 3.0: Dplenny.crypto is the decentralized website of the Dapp that runs on a blockchain domain and provides the user interface via decentralized storage using IPFS. Go to Plenny.link to find out how to access blockchain domains and decentralized websites.

4.2 Initial Verification

When the DLSP module is configured and running, you must connect with the Dapp. Go to dplenny.crypto and log in with your configured Ethereum wallet.

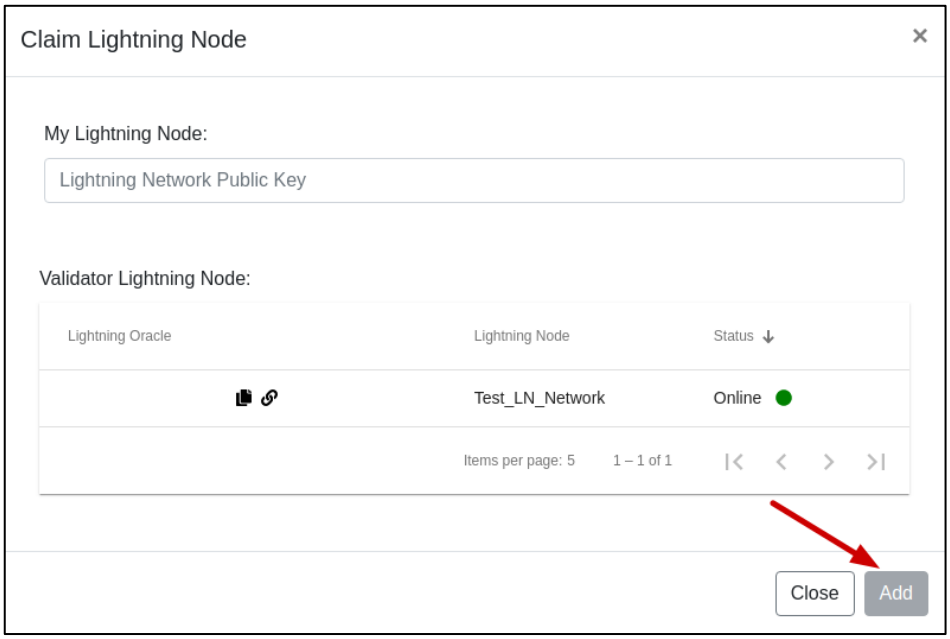
Step 1: Plenny Dapp Lightning Node Page

1. For initial verification, go to dplenny.crypto/#/verify-lightning-nodes
2. Click "Claim Lightning Node"



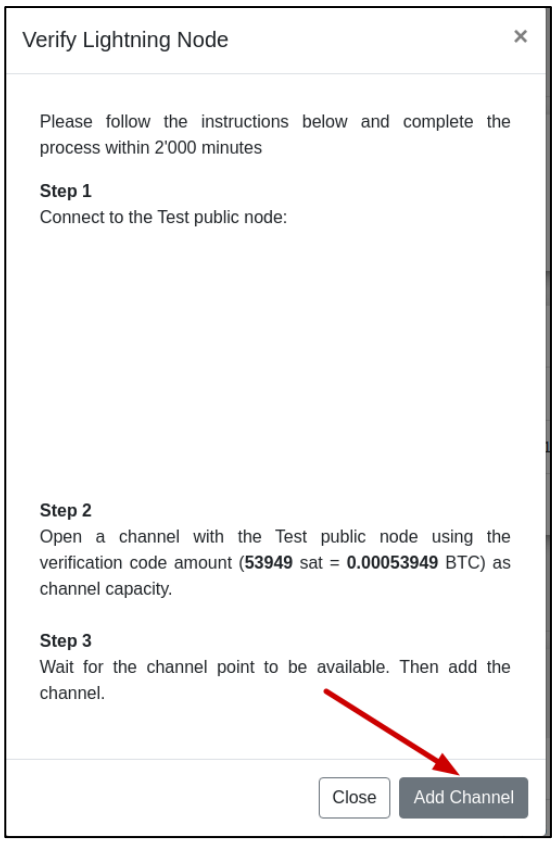
Step 2: Claim your Lightning Node

1. Insert the Public Key
2. Select Validator Lightning Node from the table
3. Click “Add” and confirm the transaction (Ethereum wallet)



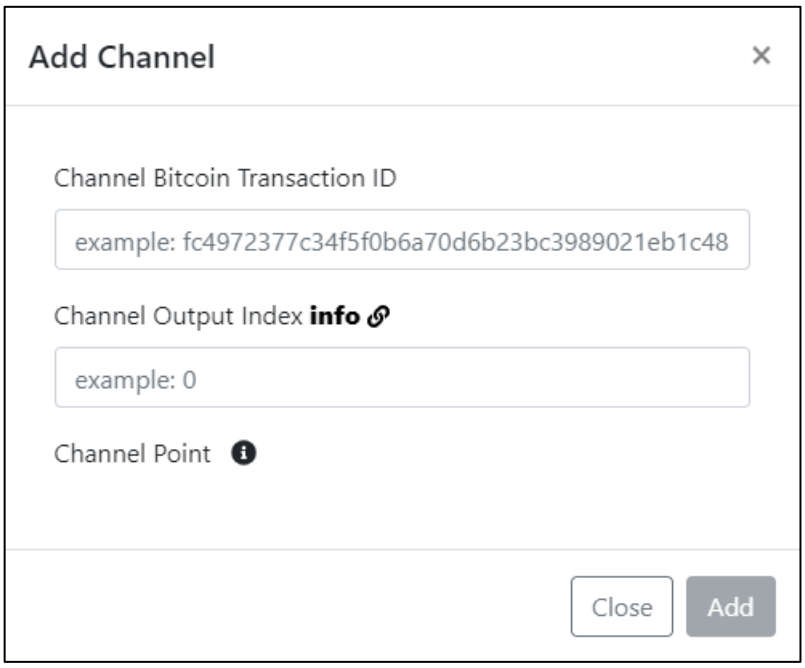
Step 3: Verify your Lightning Node

1. Open a payment channel with the given capacity to the Lightning Node which operates as a Lightning Oracle Validator (LORCA)
2. Click “Add Channel”



Step 4: Add Channel

1. Insert your Payment Channel Transaction ID and Channel Output Index
2. Click “Add” and confirm the transaction (Ethereum wallet)



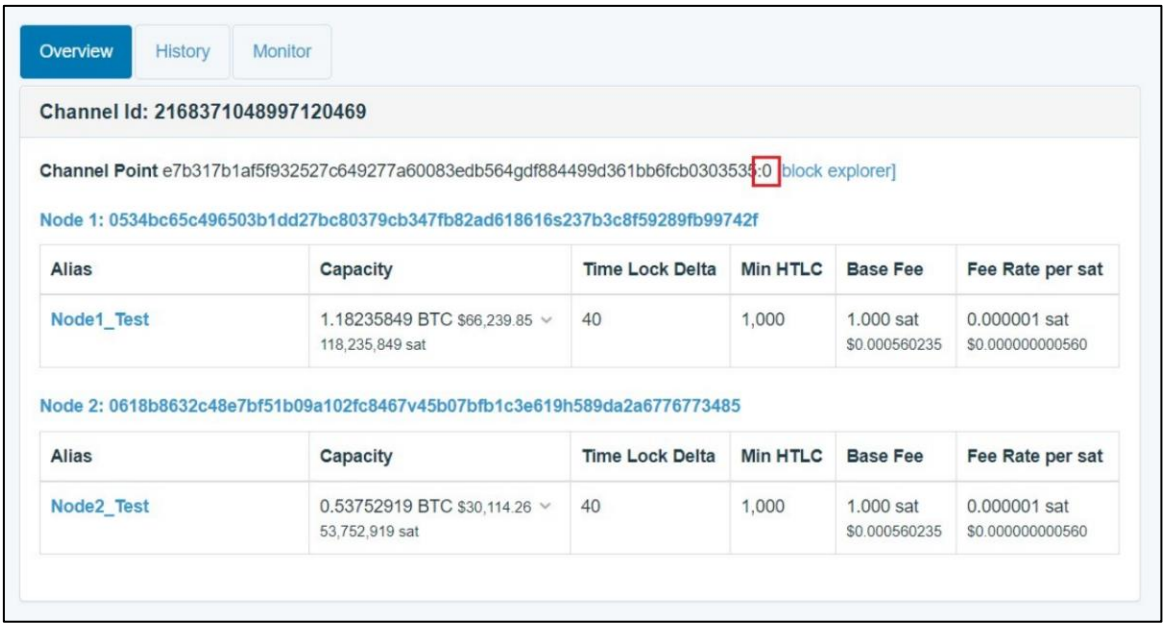
The Channel Point is the combination of the Channel Bitcoin Transaction ID and the Channel Output Index. This data comes from the Bitcoin blockchain and can be found on any blockchain explorer.

Q: How do I find the Channel Output Index?

A: There are several ways to find the Channel Output Index. You must first get the transaction ID of the channel opening transaction. A quick and easy way to find the necessary data is to visit a Lightning Network directory. Search for more details by entering your transaction ID.

In case of 1ML.com, check the record of the Channel Point. The Channel Output Index is the last number after the colon. It is either 0 or 1.

See the example as mentioned in the FAQs on plenny.crypto here:



Alternatively, you can go to a blockchain explorer like https://blockstream.info/ and enter the transaction ID. Identify the correct transaction output based on the amount of BTC. The associated "#" value is the Channel Output Index.

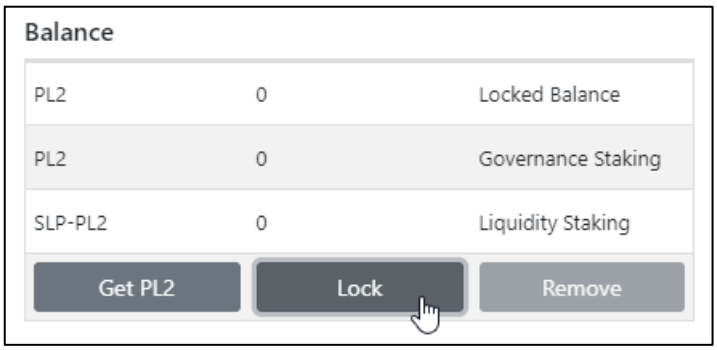
In the following example, the channel was opened with 0.0054 BTC and the Channel Output Index is "0"

5 Launch the DLSP Module on the Dapp

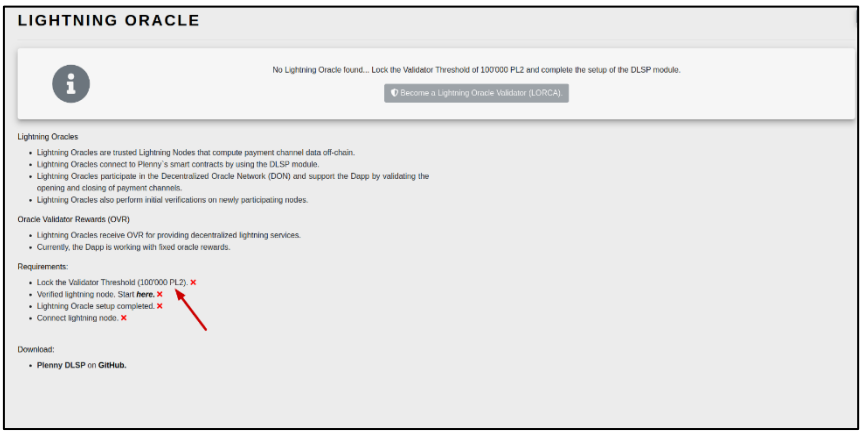
Once your Lightning Node is verified via the Dapp, you can choose to become a Lightning Oracle, a Liquidity Maker, or both.

5.1 Launch Lightning Oracle

1. Lock the Validator Threshold (e.g. 100'000 PL2) on the right sidebar under "Locked Balance"



2. Go to dplenny.crypto/#/lightning-oracle

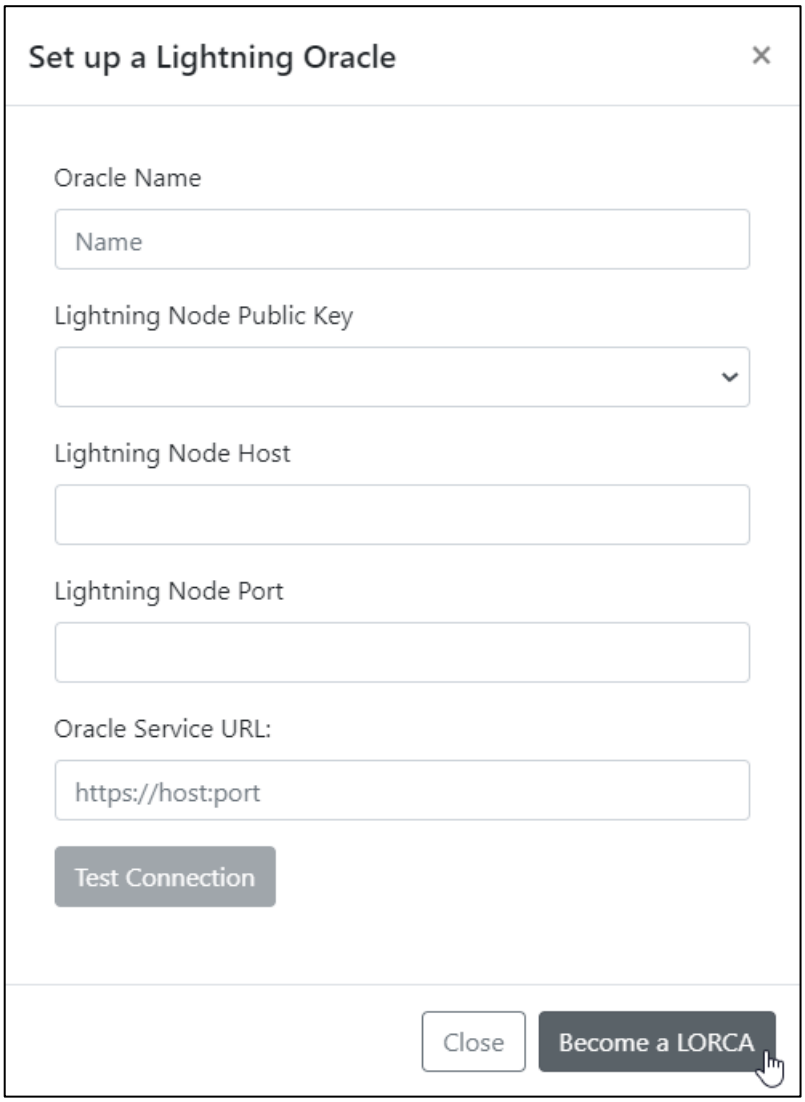


3. Click on "Become a Lightning Oracle Validator (LORCA)"
4. Add and save the details as follows:

   - *Oracle Name* = Give a Name. This is what will be shown to other users.
   - *Lightning Node Public Key* = Select your Lightning Node. Attn: Your node will only appear if the initial verification has been completed.

- *Lightning Node Host* and *Port* = Public Node IP address.
- *Oracle Service URL* = Secure DLSP module endpoint (https).

Example:

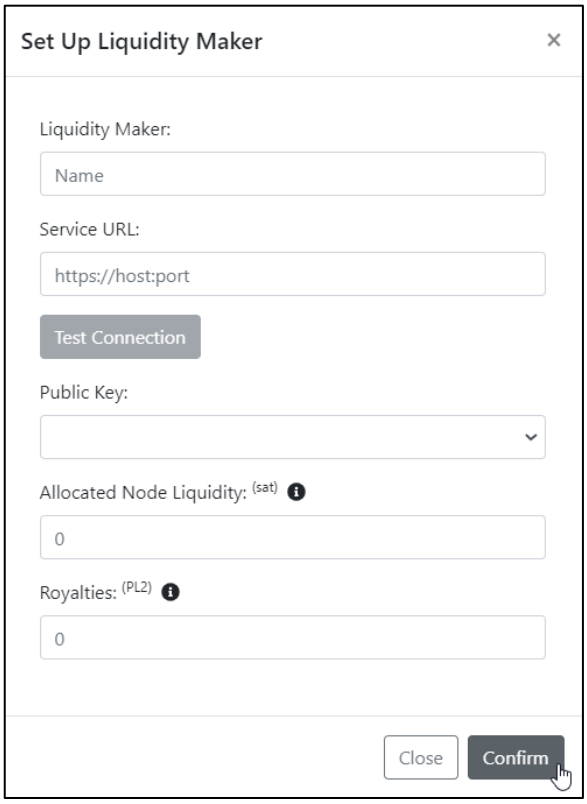


5.2 Launch Liquidity Maker

1. Go to dplenny.crypto/#/lightning-ocean
2. Click on "Become Liquidity Maker"
3. Set up the Liquidity Maker as follows:

   - *Liquidity Maker* = Give a Name. This is what will be shown to other users.
   - *Service URL* = Secure DLSP module endpoint (https).
   - *Public Key* = Select your Lightning Node.
     Attn: Your node will only appear if the initial verification has been completed.
   - *Allocated Node Liquidity* = Maximum amount of bitcoins (Satoshi) you want to allocate to the payment channel to provide inbound capacity to Liquidity Takers.
   - *Royalties* = Price you would like to get from a Liquidity Taker per Satoshi in PL2.

   Example:



Part II Operational Concept

6 Operations

6.1 Interoperability

The DLSP-module facilitates interoperability and allows Lightning Nodes to perform transactions via the Bitcoin and Ethereum blockchain. To interact across both blockchains, users run the DLSP module. This add-on module supports the provision of lightning services on the capacity market via the Dapp as well as the provision of validation services via the Decentralized Oracle Network (DON). The multifunctional open-source software module computes transaction data locally. Computation functions log off-chain data from payment channels with on-chain data sources, thereby linking Lightning Nodes with the Dapp and the currency pair sat/PL2.

6.2 NCCR-mechanism

The DLSP module validates and provides data to the smart contracts of the Dapp using a key concept called Non-Custodial Channel Rewards (NCCR). The NCCR-mechanism is specifically designed for Lightning Nodes to incentivize participation on the Lightning Network via Plenny.

The NCCR-mechanism works through the PlennyCoordinator and PlennyOracleValidator smart contracts. The coordinator contract links the Bitcoin Lightning Network and the Ethereum blockchain. It stores payment channel data and allows users to provide info about their Lightning Nodes and channels and manages the specific Baseline Reward and the general Channel Rewards. PlennyOracleValidator runs channel validations (for opening and closing) and contains the logic for reaching consensus among the Lightning Oracle Validators (a.k.a. “LORCA”) that participate in the Decentralized Oracle Network (DON). This smart contract uses data from the Plenny DLSP module. In this role, Lightning Oracles compute transaction data off-chain and transmit it to the smart contract.

In this process, channel capacity is logged in sat and linked to PL2. Logging channel capacity on the Dapp is not an allocation of money, but a validation of implied collateral in publicly verifiable payment channels, resulting in rewards for sharing transaction data over payment communication channels. The concept provides a unique procedure to leverage bitcoin and allows Lightning Nodes to participate in the Ethereum ecosystem. Sats always remain in payment channels under the control of the associated Lightning Nodes. Specifically, channel capacity is not collateralized by holding sats in Bitcoin smart contracts. Instead, the Royalties and Channel Rewards due in PL2s are locked in an Ethereum smart contract. The NCCR mechanism works similarly to virtual payment vouchers, but instead of tying the bitcoin value of the capacity in the channel to sat/PL2, only the transaction costs, licensing fees and rewards associated with the lightning services are tokenized.

Currently, the Dapp only supports public payment channels.

7 Capacity Market

The capacity market is branded as Lightning Ocean (LOC). It is decentralized, permission-less, and non-custodial. Plenny's capacity market for payment channels facilitates the licensing of inbound capacity and enables Lightning Nodes to collect Royalties and earn LOC Channel Rewards in PL2.

The roles of the participants in the capacity market are clearly defined: Liquidity Makers act as licensors to share their capacity with Liquidity Takers who act as licensees and pay Royalties for the same.

To clarify, the capacity market is not an exchange with a matched order book. Instead, it is a decentralized peer-to-peer marketplace. Makers post their outbound capacity to provide inbound capacity to Takers in sat and quote the Royalties in PL2. Takers can opt to take the offer or not.

7.1 Liquidity Maker (LM)

The role of the Liquidity Maker enables Lightning Nodes to participate in the capacity market while acting independently on the LN. Makers use the DLSP module to provide non-custodial inbound capacity to other Lightning Nodes (i.e. receiving payments limited to remote balance). LMs earn Royalties for licensing transaction data services via payment communication channels on the one hand and receive Channel Rewards from Plenny on the other.

Royalties are customizable and indicated in PL2 relative to bitcoin, but there is no fixed formula per sat as Makers are free to set the licensing fee and negotiate with Takers according to market conditions.

When specifying Royalties, Makers take into account the duration of the payment channel when providing inbound capacity, as the license fees are paid on a usage basis. Takers only pay for consumption for as long as they use the lightning services offered.

PL2 is the payment method on the Dapp while sat is used in parallel to provide payment channel capacity. In the process, LMs transact with Liquidity Takers directly using both, sat and PL2.

In summary, the capacity market works as follows:

- Lightning Nodes act as Liquidity Makers or Takers.
- The cost for inbound capacity is reflected in the Royalties (i.e. licensing fees paid by Takers).
- Royalties are subject to market demand and are therefore negotiable: Makers are free to post an asking price, Takers can accept or offer a bid price.
- Royalties are paid in PL2 serving as the payment token of the capacity market.
- Inbound capacity is provided in sat over the LN.
- Makers specify both of their rates independently, including the regular transaction fees in sat for providing inbound capacity over the LN and the Royalties in PL2 for rendering services to Takers over the capacity market.
- In addition to earning Royalties over the capacity market, Makers receive the Baseline Reward and Channel Rewards from the Treasury HODL.
- Makers receive the first payment (i.e. Fixed Royalty) right after the channel has been opened (to partially offset the costs of the metatransaction passed on by the Taker).
- Channel Rewards are locked for at least as long as the payment channel is open.
- Makers realize earnings in PL2 when the total Royalties and the Channel Rewards are collected.

7.2 Channel Licensing

In legal terms, channel licensing is an arrangement through which a licensor licenses transaction data services via payment communication channels to a licensee.

Using Plenny, Lightning Nodes provide connection access to payment communication channels which allow for sharing transaction data. Transaction data consist of information about node ID, channel ID, channel capacity, and licensing fees.

The arrangement among users concerns the right to use channel capacity but does not include custody or transfer of third-party funds. Channel licensing occurs on a non-custodial basis and is based on P2P and P2C transactions between Lightning Nodes using the Lightning Network for channel capacity and their Ethereum wallet for payments in PL2.

As Royalties are invoiced in PL2, Plenny's licensing model for inbound capacity is technically understood to represent a tokenized licensing agreement for transaction data services via payment communication channels.

In terms of compliance, Lightning Nodes providing inbound capacity are software service providers but do not qualify as financial intermediaries subject to regulation.

7.3 Royalty Calculation

In general, it is assumed that there are costs associated with licensing transaction data services through payment communication channels. The fully absorbed costs relate to expenditures for the maintenance of payment channels, including the capital cost for inbound capacity in sat and cost of capital in PL2, LN transaction fees, the gas fees in ETH, the expenses for hardware and software, other overhead, and the net margin of the Lightning Node.

Ultimately, these costs are covered by the LM. Consequently, licensors must add their net margin when licensing inbound capacity through the capacity market. The recommended net margin to be included in the Royalties is +20% in addition to the primary costs. Liquidity Makers may perform their price calculation independently and set the licensing fees in PL2 at their own discretion.

7.4 LOC Channel Rewards

Besides receiving Royalties from LTs, Liquidity Makers receive LOC Channel Rewards from Plenny for opening payment channels and logging the liquidity of Lightning Nodes over the capacity market.

The General Channel Rewards Formula is applied to LOC Channel Rewards. The formula calculates Channel Rewards for logging outbound capacity (i.e. sending payments limited to local balance). Simply put, Lightning Nodes receive Channel Rewards in proportion to the capacity (i.e. amount of sat) and relative to the duration of the payment channel, depending on Plenny's reserves. As a result, the value of LOC Channel Rewards is a fixed amount or a percentage amount depending on which is lower.

In summary, LOC Channel Rewards work as follows:

- Liquidity Makers receive LOC Channel Rewards automatically through logging payment channel capacity by using the DLSP module.
- The general reward threshold is set to X sat (e.g. 500,000), meaning Channel Rewards require this minimum amount of capacity in payment channels before LOC Channel Rewards can be generated.
- The decisive factor in the general formula is the capacity multiplier. By using a capacity multiplier, higher capacities receive above-average rewards. This exponential growth calculation is intended to favor the participation of high-volume payment channels provided by stable nodes. Channel Rewards grow exponentially depending on the outbound capacity in sat.
- In addition, Liquidity Makers receive the Baseline Reward. This minimum reward serves as a basic compensation in addition to the LOC Channel Rewards to make up for blockchain transaction costs.

- The Baseline Reward is a fixed amount (e.g. 250 PL2) and based on estimated average exchange rates. It offsets the cost of both Ethereum transaction fee (gas) and the Bitcoin on-chain transaction fee. The fixed amount in PL2 is designed to be equal or higher than the estimated total cost of gas and bitcoin fees and is supposed to include a +20% bonus. The bonus covers currency risk to which the transaction-related costs of the Lightning Nodes are exposed.

7.5 Crypto Cash Management

Crypto cash management is carried out by the Liquidity Maker itself. Based on the P2P-principle, users control their wallets themselves, allocate and remove cryptocurrency in bitcoin, ether, and PL2 independently, and set their own transaction fees in sat on their Lightning Nodes.

7.6 Semi-Automated Operations

Liquidity Makers operate with little manual intervention as long as they maintain a sufficient balance of sat on their Lightning Node to open payment channels and keep enough ETH in their Ethereum wallet for transactions.

The DLSP module runs continuously, computing transaction data off-chain and transmitting transaction details to the Dapp automatically. The tool assists the Maker in managing operations via Plenny 24/7.

There is no need for opening the channel manually on the Lightning Node when the Liquidity Taker (LT) requests capacity. This task is done by the DLSP module automatically. It is also not necessary to repost the offer after closing a channel as it will remain publicly listed until the maximum sat balance of the offer is reached.

The offer stays valid on the capacity market and enables Liquidity Makers to automatically serve further requests and license capacities

to multiple LTs via different payment channels in the Lightning Network. Makers only need to regularly check whether the maximum performance period still matches the cost-benefit ratio.

However, Makers can manually reset the details at any time by updating Royalties, reducing or increasing capacity, and removing and reposting the offer. For instance, profitless channels can be closed manually or automatically depending on the configuration.

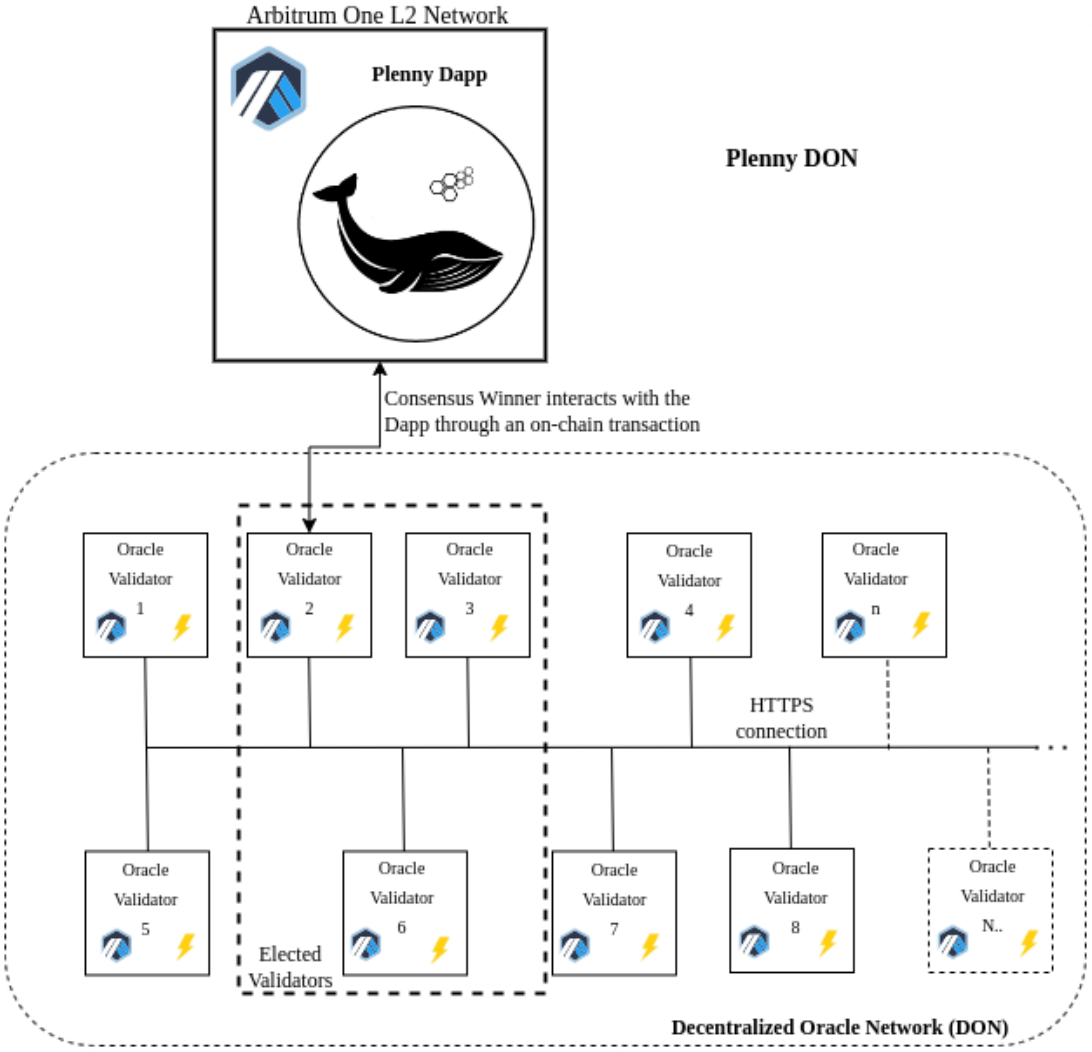
For this use case, manual collection is required to get tokens into the user's wallet. Apart from that, most operational tasks on the capacity market are fully automated. The LM simply needs to monitor current market conditions over the Dapp to adjust Royalties and capacity as needed.

8 Decentralized Oracle Network (DON)

Plenny provides a sophisticated trust-building mechanism using a Decentralized Oracle Network. Participating Lightning Oracle Validators serve as a chain link to connect Lightning Nodes to the Plenny network that links off-chain data from Lightning Nodes with on-chain data sources.

8.1 DON Diagram

The architecture of the oracle network is based on Web 3.0 as well as Web 2.0 technologies interacting on-chain and off-chain.



8.2 Lightning Oracle Validator (LORCA)

Lightning Oracle Validators validate the opening and closing of payment channels. In general, LORCAs safeguard the connectivity of Lightning Nodes and payment channels. In the process, they monitor the channel activities on the Dapp, including the capacity market, and in return they are incentivized for consensus building on the Decentralized Oracle Network (DON).

8.3 Consensus Mechanism

Both the payment channel consensus and the network consensus between the Lightning Oracles are reached on the basis of Proof of Stake (PoS). All Lightning Nodes working as oracle validators and participating in this process form the DON of the Dapp. This consensus mechanism ensures the functionality and security of Plenny.

The DON is neither a lightning hub nor a sidechain. It is an independent network interconnected with the LN and the Ethereum ecosystem. Off-chain computations are collected by oracle validators in a cryptographically secured manner and transmitted on-chain to Plenny's smart contracts. The consensus mechanism ensures data integrity and mitigates evidently valueless information from entering the DON.

In detail, the validation process to reach consensus works as follows:

- All elected validators are linked to each other via the DLSP module to share data and confirm consensus over HTTPS connections, which enables communication among oracles off-chain.
- When a new channel is opened or closed, the elected validators try to obtain enough cryptographic signatures for reaching consensus.
- The oracles get the data from the Bitcoin Lightning Network.

- The oracles compete with each other to obtain a sufficient number of signatures from other validators.
- The signature is a hash key of the channel info and channel data signed with the oracle's private key. They are generated off-chain using the DLSP module.
- The validators' signatures are matched against the channel data already stored in the smart contract. If the signatures are valid, the opening or closing of the channel is confirmed.
- The validator who obtains enough cryptographic signatures first is the consensus winner and authorized to post the validation data, confirming consensus on-chain on the Ethereum blockchain.

8.4 Leaderless Consensus

The DON uses a leaderless consensus mechanism. It includes an integrated automatic failover mechanism to systematically maintain operational effectiveness. The failover mechanism mitigates downtime of oracle validators, ensuring off-chain data is submitted on-chain continuously.

8.5 Oracle Validator Rewards (OVR)

LORCAs receive Oracle Validator Rewards for the validation of opening and closing payment channels. The compensation model for oracle validators creates permanent incentive circles and accounts for participants' costs of providing lightning services.

In detail, the reward scheme for oracle validators works as follows:

- LORCAs benefit from a specific remuneration scheme different than Channel Rewards for regular Lightning Nodes.
- LORCAs receive OVR for the initial verification of Lightning Nodes according to the OVR transaction calculation.
- OVRs are distributed automatically on a transaction-basis once the validation is successfully completed. This means OVRs are

given by the Treasury HODL as soon as the oracle validator has submitted the data to confirm the consensus on-chain. This event happens within the channel opening or closing validation transaction. Once the consensus is reached and the final state of the channel is confirmed, rewards are distributed.

- OVR is due as soon as consensus is reached and the final state of the channel is confirmed, ensuring LORCAs are compensated in real-time and not exposed to currency risk.

8.6 OVR Formula

The rewards due are calculated on a transaction basis as follows:

- OVR is either a minimum fixed number of token (F) or a percentage (P) of the amount reserved in the Treasury HODL (depending on which is lower).

Based on the formula for reserve sustainability, the reward percentage is multiplied by the Treasury HODL reserves and divided by 100. If the result of this calculation is lower than the fixed amount, the percentage is applied, otherwise the fixed amount is used.

8.7 OVR Distribution Formula

In general, consensus winners earn significantly more because they pay the gas fee for submitting data on-chain and thus have higher expenses. The distribution of Oracle Validation Rewards (OVR) works as follows:

- OVR = X% of the smart contract controlling consensus.
- The reward of the functions is P (percentage) of the rewards.
- The oracle validator who submits the data to confirm consensus first on-chain earns (≈60%) of the OVR.
- The other oracles who signed the data off-chain share the remaining rewards (≈40%).

Lightning Oracles compete against each other on a transaction basis. The oracle who correctly validates the fastest wins and earns the highest amount of rewards. The others confirm the validation, and the remaining rewards are shared proportionately among all.

8.8 Validation Cycle

The validation cycle lasts for 1 week (e.g. approximate corresponding block period is ≈45,500 blocks). Only elected Lightning Oracles provide validation services, with multiple oracles ensuring data integrity. For each validation cycle, a group of validators (e.g. 3 to ∞) is elected.

- Maximum validators during the experimental phase (i.e. BETA release): ≈100 Lightning Oracles.
- Maximum number of validators in later versions: Infinite (although the gas limit on L2 suggests utilizing a limited number of validators for the early versions of the DON).

8.9 Oracle Election Trigger

This community-driven trigger initiates a new oracle election. The end of the validation cycle (and the beginning of the new one) is periodically triggered by users. Any Ethereum-user can trigger the new election cycle and earn rewards as follows:

- Users trigger the new election cycle manually at their own discretion (i.e. at the earliest after X blocks), and as a result the trigger user is incentivized with the Election Trigger User Reward (ETUR), which is a fixed reward.

8.10 Oracle Election

The oracle election consists of several distinct steps as described below:

- Election of new validators from all the potential LORCA candidates based on their score.

- The election is based on a snapshot of the current validator score using the following criteria: number of PL2 locked and the number of OVR earned.
- The community-driven Oracle Election Trigger function can be called at most once each week (e.g. ≈45500 blocks).
- If no election cycle is triggered manually, the DON continues with the previously elected oracle validators.
- After the validators have been elected, their balances of PL2 token locked and earned are snapshotted and used throughout the whole validation cycle.

Lightning Oracles may adjust their locked balance during the current validation cycle, but this adjustment will not take effect until the beginning of the upcoming validation cycle to allow oracles enough time to review and adjust their figures if necessary. Both the number of validators and the duration of the validation cycle are subject to governance voting.

8.11 Operative Use

Lightning Oracle Validators operate with little manual intervention as long as they lock the validator threshold and keep enough ETH in their Ethereum wallet for transactions. The DLSP module runs continuously. It assists Lightning Oracles in managing operations on the Dapp around the clock. This use case is recognized by the Plenny DLSP module, which supports automated operations like subscribing to events and trigger functions.

9 Miscellaneous

9.1 FAQs

Q: How does a Liquidity Maker earn money?

A: Liquidity Makers earn money for licensing transaction data services via payment communication channels. Licensing inbound capacity enables Liquidity Makers to earn Royalties and receive Channel Rewards.

Q: How does a Lightning Oracle earn money?

A: Lightning Oracles earn money for validating the opening and closing of payment channels on the Lightning Network.

Q: Where is the Plenny DLSP module available for download?

A: You can download it on GitHub at https://github.com/PlennyPL2.

9.2 Cost-Benefit-Analysis

Lightning Nodes using the Plenny DLSP module can generate income from token rewards in addition to sat income. Reduced capital cost, lower expenses, and higher income are achieved through increased volume provided via the additional liquidity of the Ethereum ecosystem.

The estimated income depends on the use case. Sample calculations show the following cost-benefit analysis:*

Use Case Capacity Market:

If a Liquidity Maker provides 3.9M sat of inbound capacity for 250 days to a Liquidity Taker, the cost-benefit analysis looks as follows:

+5,436.78 PL2 Royalties and Channel Rewards

- Fixed Royalty: 50 PL2.
- Customizable Royalty: X PL2 (this licensing fee is freely determined by the Liquidity Maker).
- LOC Channel Rewards: Baseline Reward of 250 PL2 + 5,136.78 PL2 = 5,386.78 PL2.

-661.25 PL2 Expenditure (gas fee and other cost)

- 1 transaction with gas fee: USD 3.60 and 1 Bitcoin on-chain transaction: USD 3.00 = USD 6.60 = 660 PL2.
- Maker Fee of Fixed Royalty Amount and Customizable Royalty Amount: 2.5% = 1.25 PL2 + X PL2 = 1.25 PL2.

Minimum Estimated Income after 250 days = +4,775.53 PL2

Use Case Decentralized Oracle Network (DON):

If a Lightning Oracle validates the opening and closing of a payment channel of 3.9M sat, the cost-benefit analysis looks as follows:

+4,800.00 PL2 Oracle Validator Rewards (OVR)

- OVR for the Consensus Winner: 2 validations: 2 x 2,400 PL2 = 4,800 PL2.

-1,641.00 PL2 Expenditure (gas fee and other cost)

- 2 transactions with gas fees for opening and closing of the channel: USD 16.41= 1,641 PL2.

Estimated Income per validation = +3,159.00 PL2

*The sample calculations are meant only as examples. Such calculations can vary widely depending on transaction fees, exchange rates (i.e. BTC, ETH, and PL2), and multiple DAO Governance parameters regarding reward levels, including the reserves held in the Treasury HODL. The following exchange rates and parameters used here are for illustrative purposes only: 1 PL2 = USD 0.01; 1 ETH = USD 2,300.00; TH = 250,000,000 PL2.

9.3 Release & Version History

This Operational Manual on the DLSP module refers the Plenny BETA Release of the smart contracts dated February 4, 2022 and the Plenny UI v1.0.3 dated April 21, 2022 on Dplenny.crypto. Review the current documentation on plenny.crypto to learn more about the newest features of the Dapp and the latest functionalities of the DLSP module.

9.4 OSS Licensing

Plenny grants data access through open-source software (OSS). The intellectual property released by Plenny is in the knowledge commons. The decentralized application with its native token and the DLSP module is a digital public utility maintained by a global community. Plenny serves as a shared resource for the common good of all users. The code is licensed under the MIT License for free and open-source software (open-sourced on GitHub).

9.5 Privacy

Privacy is ensured between the participants on a peer-to-peer basis. The Plenny Dapp cannot track lightning transactions from participating nodes. Lightning transactions only take place on a P2P basis and can therefore only be tracked by Lightning Nodes in their own payment channels. Using Hashed Timelock Contracts (HTLC), participating nodes cannot track the lightning transactions of others.

Lightning Nodes permit participants to access channel capacity on a non-custodial basis for connectivity and transactional purposes. Access to payments data is granted based on P2P transactions. During this data exchange, Plenny enforces fundamental data protection principals. There is no control of funds by third parties, no accounts for depositing or holding funds, and no financial intermediation taking place.

Apart from that, as a general rule, crypto experts recommend having more than one crypto wallet, including at least one that is public-facing and at least one that is private.

9.6 DYOR

Do your own research (DYOR)! This notice serves as a disclaimer. It is advisable to read, review, learn, and understand the details of Plenny before testing it.

None of the information or analyses in this manual are intended to provide a basis for an investment decision, and no specific investment recommendation is made. This manual does not constitute investment advice, nor is it an invitation to invest in any security or financial instrument. Instead, the information about Plenny in this manual serves educational purposes only.

Furthermore, Plenny relies on several random assumptions and technical prerequisites that have not been sufficiently tested to make a valid statement. As such, the models presented are only as good as their random assumptions. Any significant deviation from the input numbers would subsequently impact the outputs.

All forward-looking statements address matters that involve risks and uncertainties, and do not constitute a guarantee these results will be achieved. No statement in this manual is intended as a profit forecast.

Operating decentralized applications is considered a high-risk activity as the use cases and regulatory implications are not clear in most

jurisdictions. Any statement about the possible issuance of cryptographic tokens in this document concerns highly complex issues in technical, economic, and legal terms, which is why the stated content must be fundamentally questioned, as the information may be incorrect.

Consult legal, financial, tax, or other professional advisors if you are in any doubt. No regulatory authority has examined or approved any of the information set out in this manual. No such action has been or will be taken under the laws, regulatory requirements, or rules of any jurisdiction as Plenny does not reside in a specific jurisdiction.

The information in this manual may not be exhaustive and does not imply any elements of a contractual relationship or obligations. No part of this manual is legally binding or enforceable, nor is it meant to be.