LNP/BP Standards Association

LNP/BP v0.3 Libraries Release

for descriptor-based bitcoin wallets,
generalized lightning network, Internet2 and
RGB smart contracts

LNP/BP Standards Association
Prepared & supervised by **Dr Maxim Orlovsky, Pandora Core AG**

v0.3 releases of main libraries:

- Updated to **bitcoin 0.26** and **miniscript 5.0**
- **RGB Core** Library extracted to https://github.com/rgb-org/rgb-core
- **LNP Core** Library extracted to https://github.com/LNP-BP/lnb-core
- **Internet2** repository & crates extracted to https://github.com/internet-org/rust-internet2
- Bitcoin **descriptor wallet** library extracted to https://github.com/LNP-BP/descriptor-wallet
- Repository split into multiple crates
  (lnpbp, client_side_validation, strict_encoding, strict_encoding_derive)
- Lightning-specific message encodings for interoperability with other LN nodes
- **BOLT-7** messages support in LNP Core by *Rajarshi Maitra (@raj)*
- Refactored unified network address format & encodings (inside strict_encoding)
- Refactored deterministic bitcoin commitments (LNPBP-4)

Roadmap

v0.4 (March): completing LN

- Universal invoices
- LN routing & gossip protocol
- LNP Core & Node to be interoperable with other LN implementations

v0.5 (May): Infrastructure

- Bifrost protocol in LNP Core & Node
- Encrypted RPC communications with nodes
- Completed Tor support
- Updated & refactored BP Node
- Using BP Node alongside Electrum

- Nonfungible assets (RGB21) implementation

Pending issues

- #47 RGB-SDK: Fail to link librgb_node library on Android
  - Looking for help
- #83 (LNPBPs): Miniscript breaks legacy scripts compatibility in LNPBP-2 (including lightning network)
  - small impact: LNPBP-2 can’t be used outside of RGB context, thus no compatibility problems
  - Solutions:
    - leave as is: “upgrade” RGB LN to “miniscript” scripts
    - change LNPBP-2: use miniscript for key detection but not for key replacement
    - change LNPBP-2: do not use miniscript

Bitcoin-related libraries

LNP/BP Core: LNPBP standards

- Deterministic bitcoin commitments
  (LNPBP-1, 2, 3, 4)
- Bitcoin single-use seals
  (and blinded UTXOs)
- Short bitcoin ID
- Chain parameters
  (mainnet, testnet, signet, liquidv1, custom)
- ElGamal encryption with Secp256k1 keys
- Tagged hash extensions

Wallet Descriptors: Layer 1 stuff

- Script types
- Descriptors
- BIP32 extensions
- SLIP132
- Hashlock contracts-related types
- PSBT extensions
- Lexicographic ordering
- Feature Flags

Libraries & repositories refactoring

- Release of bitcoin 0.26 and miniscript 5.0 breaking much of APIs
- Separating mission-critical & utility parts
- Simplifying dependencies & avoiding complex version conflicts
- Improving compile times & library sizes
- Allowing more portable use of RGB primitives outside of node scope
- Allowing use of LNP/BP features outside of RGB scope
  (bitcoin wallets, LN)
- Careful review of existing codebase
- Migrating common parts to upstream repos
- Aligning peer-review & merge requirements with repo importance

Products

- Standards (expert community)
  - LNP/BP Standards
  - RGB Specification (Yellow Paper)
- Software (end users & hosting providers)
  - RGB Node (on github.com/rgb-org)
  - LNP Node
  - BP Node
- SDKs (end-user software developers)
  - RGB SDK
  - LNP SDK

Libraries

for low-level software developers

- Internet2 Libraries (multilanguage)
  - Internet2
  - Microservices
- LNP/BP Libraries (pure rust with C & WASM FFI)
  - LNP/BP Core Library
  - Bitcoin Descriptor Wallet Library
  - RGB Core Libraries (RGB + RGB20 etc schemata)
  - LNP Core Library
- Platform-specific class libraries for ECMAScript, JVM, CLR, Swift, Python, Go
  - RGB Class Libraries (used in RGB SDK)
  - LNP Class Libraries (used in LNP SDK)

Must be carefully reviewed

- LNP/BP Core Library
  - bitcoin commitments
- RGB Core Library
  - zero knowledge (bulletproofs, Pedersen commitments)
  - schema validation
  - state transition graph validation
  - virtual machine validating state evolution

Repository & libraries structure

Primitives

▨ Rust crates ⬚ Git repositories □ Products 🔵 github.com/internet2-org 🟠 🟡 github.com/LNP-BP 🔴 github.com/RGB-org

Repository & libraries structure

Wallet Descriptors Library

descriptor-wallet

slip132

psbt

bip32

miniscript

bitcoin

LNP/BP Core Lib

lnpbp

strict_encode

client_side_validation

amplify

Internet2

internet2

inet2_addr

Primitives

tor

zmq

websocket

▨ Rust crates ⬚ Git repositories □ Products ● github.com/internet2-org ● ● github.com/LNP-BP ● github.com/RGB-org

Repository & libraries structure

RGB Core Library

rgb20 rgb21 rgb22

rgb

LNP Core Library

payments factories bifrost watchtowers

lnp lnp2p

Internet2

microservices

Business logic

Wallet Descriptors Library

descriptor-wallet slip132

psbt bip32 miniscript

bitcoin

LNP/BP Core Lib

lnpbp

strict_encode client_side_validation

amplify

internet2

inet2_addr

Primitives

tor zmq websocket

▨ Rust crates ⬚ Git repositories □ Products ● github.com/internet2-org ● ● github.com/LNP-BP ● github.com/RGB-org

Repository & libraries structure

BP SDK

RGB SDK

LNP SDK

SDKs

BP Node

RGB Node

LNP Node

Nodes

Invoicing library

invoices

Internet2

RGB Core Library

rgb20

rgb21

rgb22

rgb

LNP Core Library

payments

factories

bifrost

watchtowers

lnp

lnp2p

microservices

Business logic

Wallet Descriptors Library

descriptor-wallet

slip132

LNP/BP Core Lib

lnpbp

strict_encode

client_side_validation

internet2

inet2_addr

Primitives

psbt

bip32

miniscript

bitcoin

amplify

tor

zmq

websocket

▨ Rust crates ⬚ Git repositories □ Products ● github.com/internet2-org ● ● github.com/LNP-BP ● github.com/RGB-org

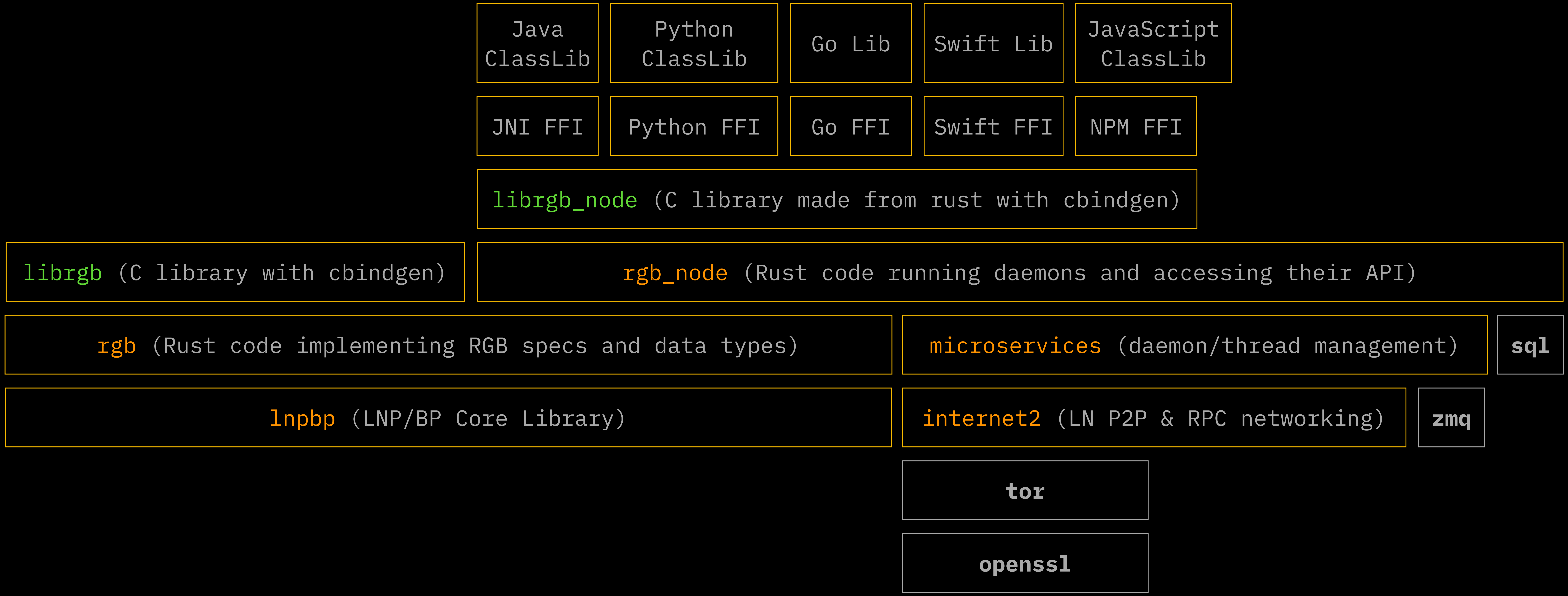
Structuring software

- **Repositories**: code contribution management with single commit/review/merge policy. May contain multiple…
  - **Packages** (crate): language-specific code which may be reused multiple times, including third-parties. Has versioning (semantic) and contributors/code reuse license. May produce multiple
    - **Artifacts** (binary, library): compilation target or composed form of package

Product: end-user installable item with a clear use purpose, non-semantic version number, user instructions and end-user license agreement (EULA). Product is shipped as a set of artifacts

	Product	Repository	Package
License type	EULA	-	FOSS
Versioning	Part of branding & marketing	-	Semantic
Used by	Users	Developer team	External developers
Docs	User guideline	Code docs	Integration docs
Access control	-	+	-
Issue tracking	Support desk	GitHub	-
Purpose	User story	Code management	Architecture abstraction

Developing software for RGB & LNP outside rust



Java ClassLib

Python ClassLib

Go Lib

Swift Lib

JavaScript ClassLib

JNI FFI

Python FFI

Go FFI

Swift FFI

NPM FFI

librgb_node (C library made from rust with cbindgen)

librgb (C library with cbindgen)

rgb_node (Rust code running daemons and accessing their API)

rgb (Rust code implementing RGB specs and data types)

microservices (daemon/thread management)

sql

lnpbp (LNP/BP Core Library)

internet2 (LN P2P & RPC networking)

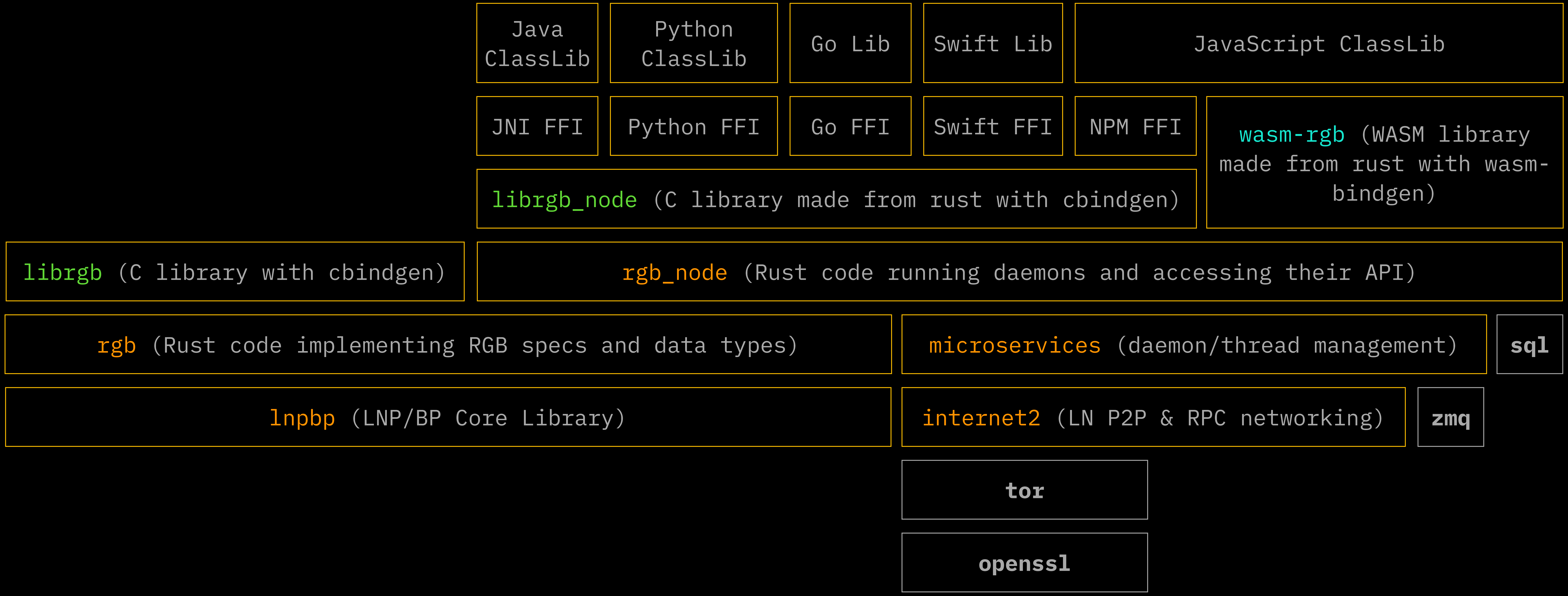
zmq

tor

openssl

Rust libraries   C libraries   **External dependencies with high compilation complexity**

Developing software for RGB & LNP outside rust



Java ClassLib | Python ClassLib | Go Lib | Swift Lib | JavaScript ClassLib

JNI FFI | Python FFI | Go FFI | Swift FFI | NPM FFI | wasm-rgb (WASM library made from rust with wasm-bindgen)

librgb_node (C library made from rust with cbindgen)

librgb (C library with cbindgen) | rgb_node (Rust code running daemons and accessing their API)

rgb (Rust code implementing RGB specs and data types) | microservices (daemon/thread management) | sql

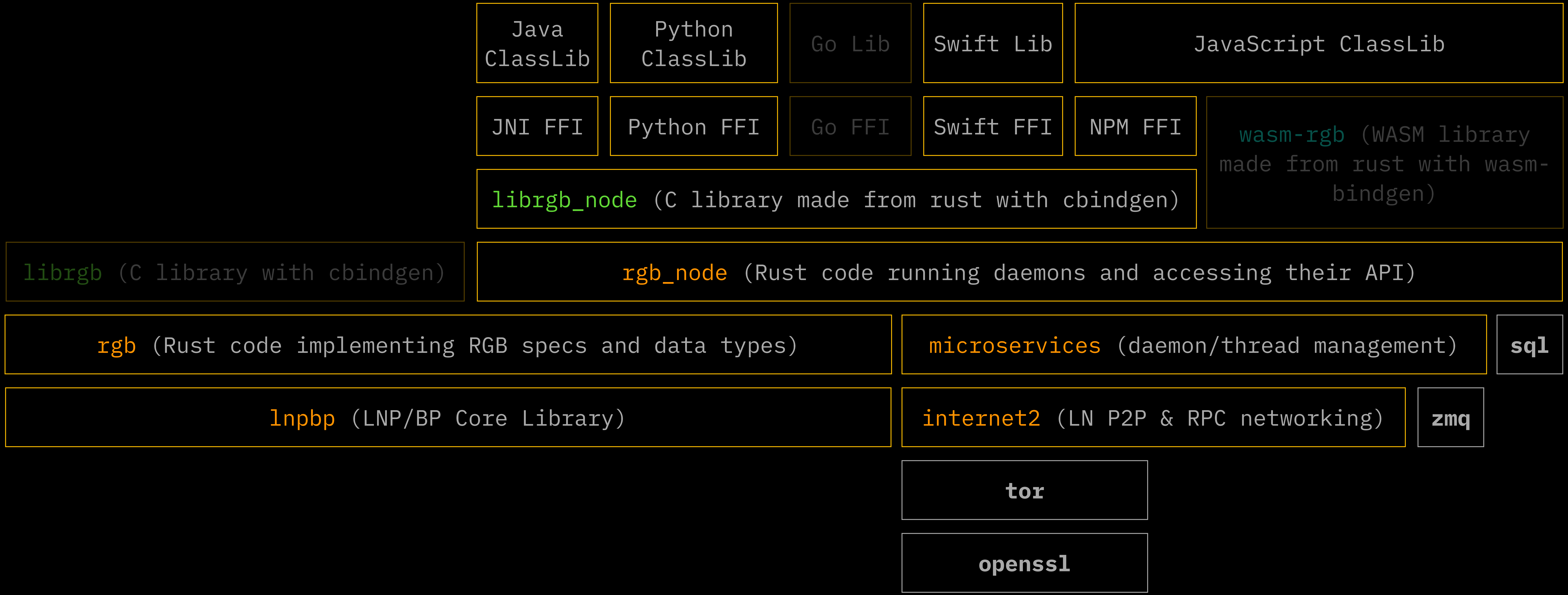
lnpbp (LNP/BP Core Library) | internet2 (LN P2P & RPC networking) | zmq

tor

openssl

Rust libraries   C libraries   **External dependencies with high compilation complexity**

Developing software for RGB & LNP outside rust



Java ClassLib

Python ClassLib

Go Lib

Swift Lib

JavaScript ClassLib

JNI FFI

Python FFI

Go FFI

Swift FFI

NPM FFI

wasm-rgb (WASM library made from rust with wasm-bindgen)

librgb_node (C library made from rust with cbindgen)

librgb (C library with cbindgen)

rgb_node (Rust code running daemons and accessing their API)

rgb (Rust code implementing RGB specs and data types)

microservices (daemon/thread management)

sql

lnpbp (LNP/BP Core Library)

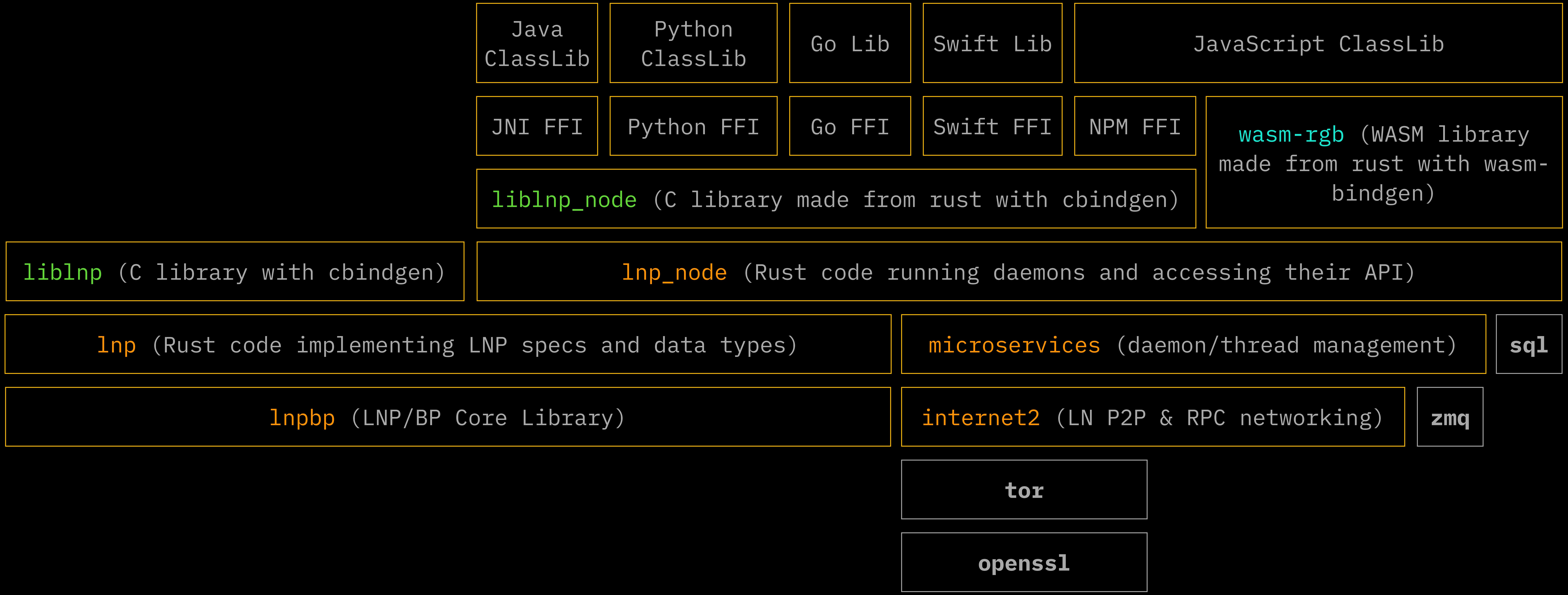
internet2 (LN P2P & RPC networking)

zmq

tor

openssl

Developing software for RGB & LNP outside rust



Java ClassLib | Python ClassLib | Go Lib | Swift Lib | JavaScript ClassLib

JNI FFI | Python FFI | Go FFI | Swift FFI | NPM FFI | wasm-rgb (WASM library made from rust with wasm-bindgen)

liblnp_node (C library made from rust with cbindgen)

liblnp (C library with cbindgen) | lnp_node (Rust code running daemons and accessing their API)

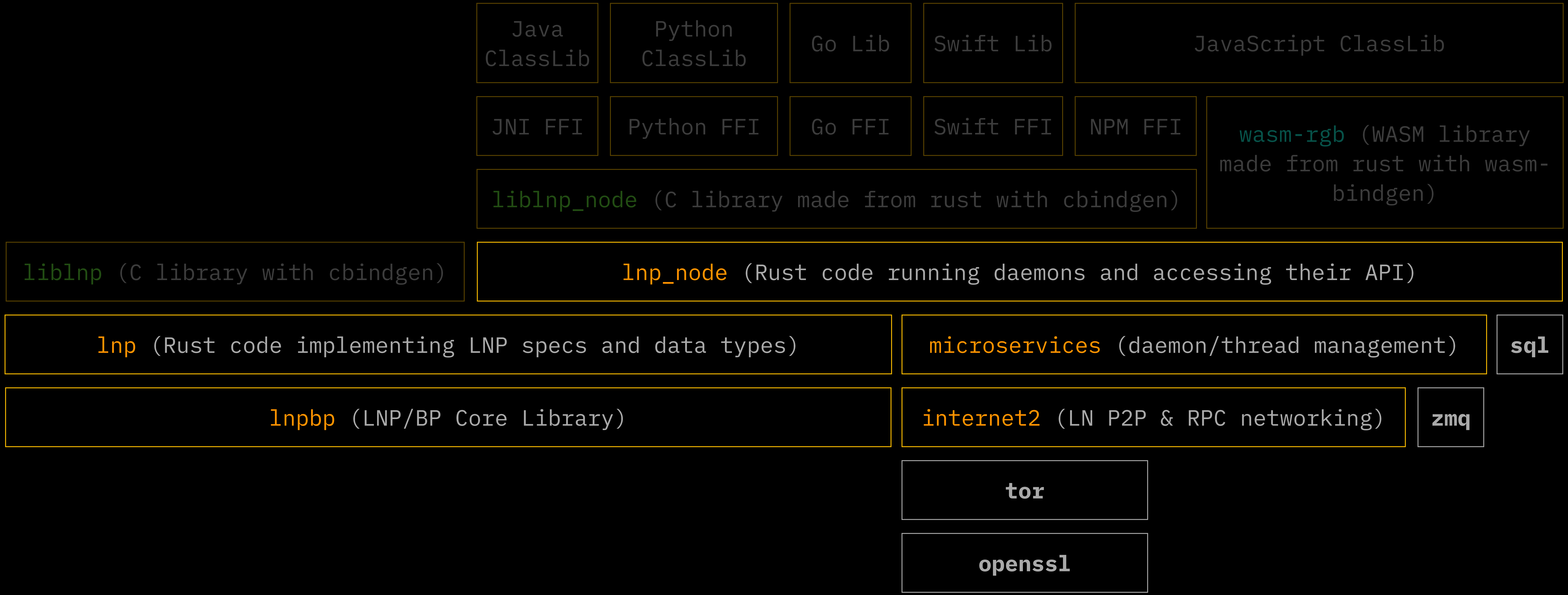
lnp (Rust code implementing LNP specs and data types) | microservices (daemon/thread management) | sql

lnpbp (LNP/BP Core Library) | internet2 (LN P2P & RPC networking) | zmq

tor

openssl

Developing software for RGB & LNP outside rust



Java ClassLib

Python ClassLib

Go Lib

Swift Lib

JavaScript ClassLib

JNI FFI

Python FFI

Go FFI

Swift FFI

NPM FFI

wasm-rgb (WASM library made from rust with wasm-bindgen)

liblnp_node (C library made from rust with cbindgen)

liblnp (C library with cbindgen)

lnp_node (Rust code running daemons and accessing their API)

lnp (Rust code implementing LNP specs and data types)

microservices (daemon/thread management)

sql

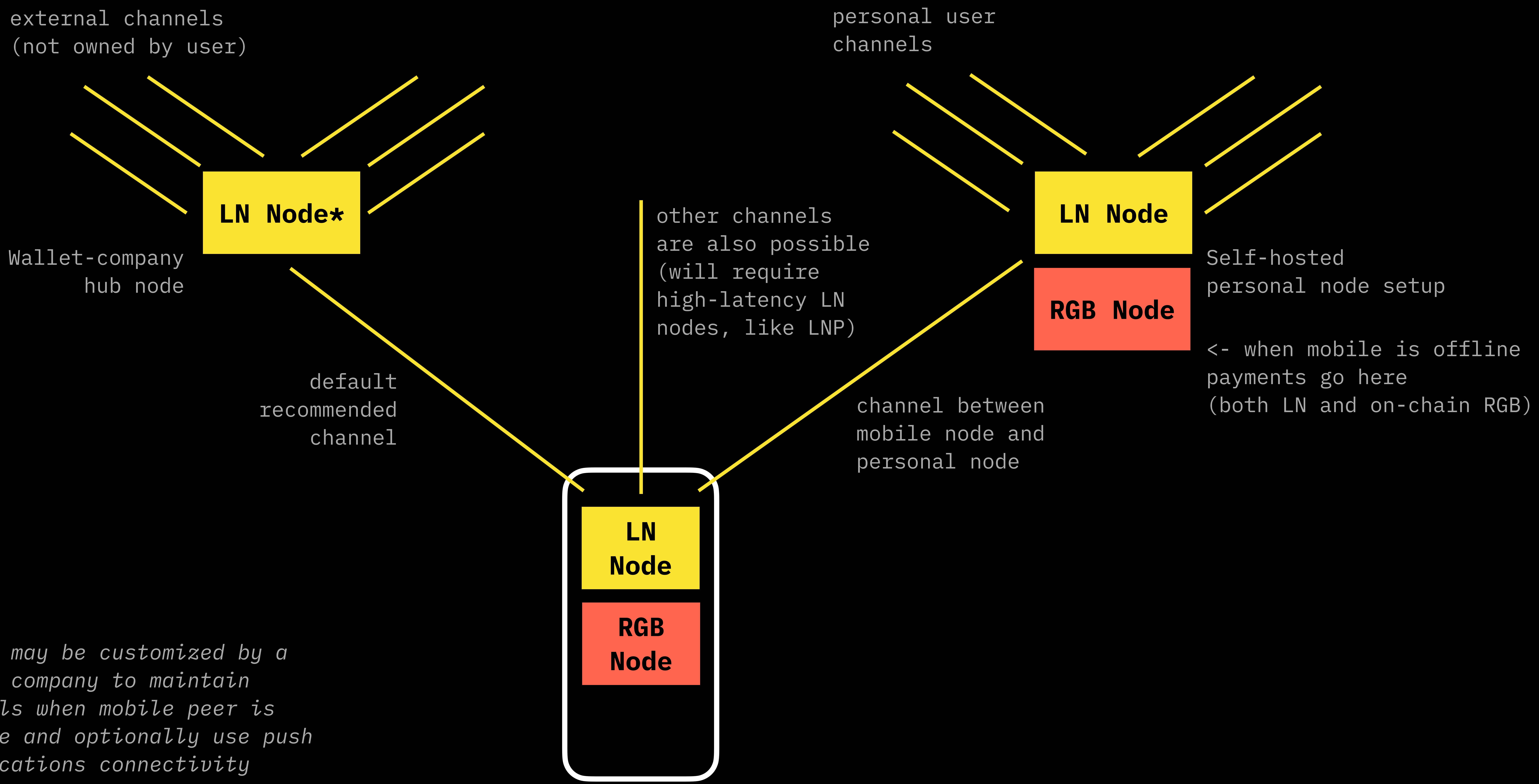
lnpbp (LNP/BP Core Library)

internet2 (LN P2P & RPC networking)

zmq

tor

openssl



external channels
(not owned by user)

personal user
channels

LN Node*

Wallet-company
hub node

other channels
are also possible
(will require
high-latency LN
nodes, like LNP)

LN Node

RGB Node

Self-hosted
personal node setup

<- when mobile is offline
payments go here
(both LN and on-chain RGB)

default
recommended
channel

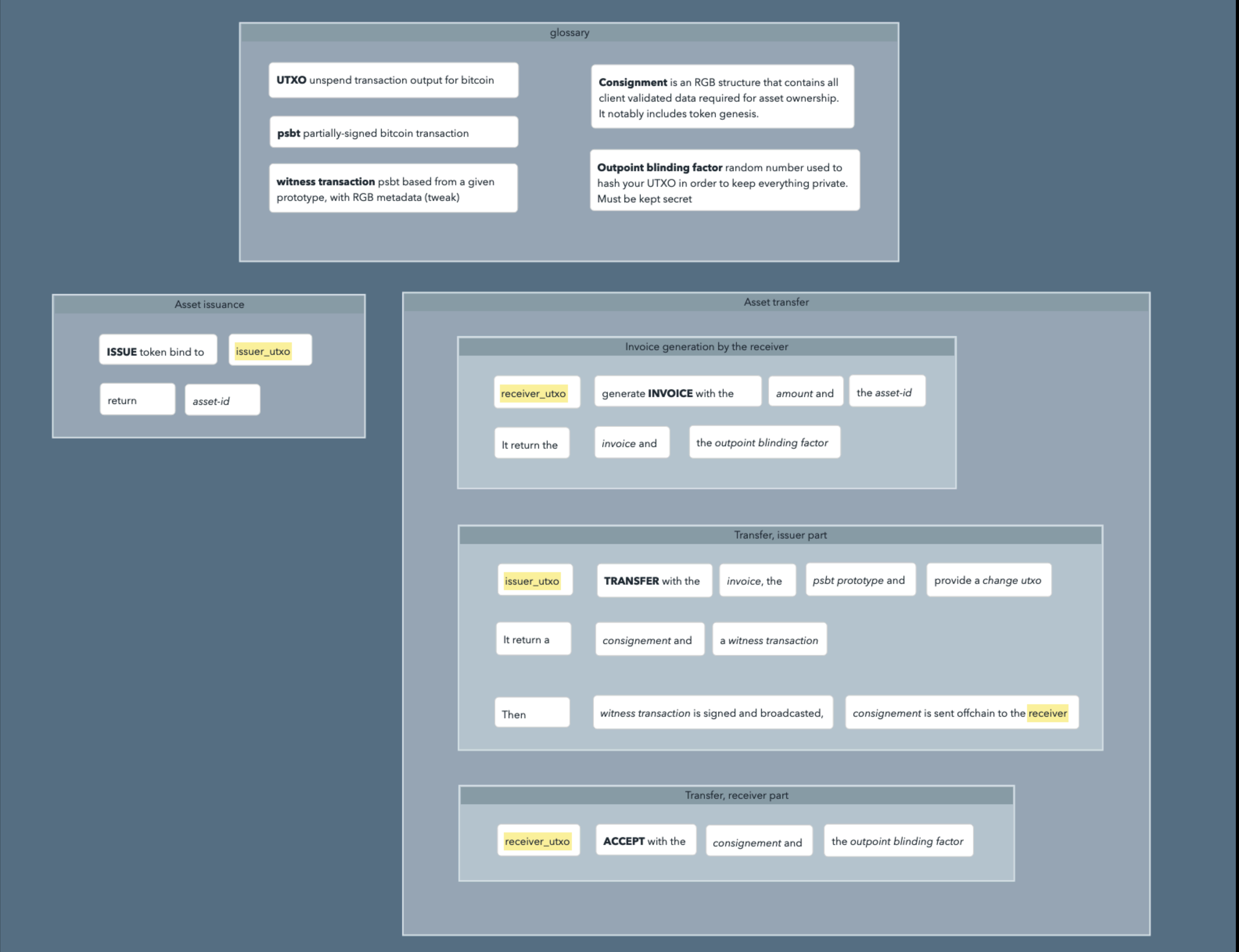
channel between
mobile node and
personal node

LN
Node

RGB
Node

** Node may be customized by a wallet company to maintain channels when mobile peer is offline and optionally use push notifications connectivity*

Schema of RGB workflows by @kipit



glossary

UTXO unspend transaction output for bitcoin

psbt partially-signed bitcoin transaction

witness transaction psbt based from a given prototype, with RGB metadata (tweak)

Consignment is an RGB structure that contains all client validated data required for asset ownership. It notably includes token genesis.

Outpoint blinding factor random number used to hash your UTXO in order to keep everything private. Must be kept secret

Asset issuance

ISSUE token bind to issuer_utxo

return *asset-id*

Asset transfer

Invoice generation by the receiver

receiver_utxo generate **INVOICE** with the *amount* and the *asset-id*

It return the *invoice* and the *outpoint blinding factor*

Transfer, issuer part

issuer_utxo **TRANSFER** with the *invoice*, the *psbt prototype* and provide a *change utxo*

It return a *consignement* and a *witness transaction*

Then *witness transaction* is signed and broadcasted, *consignement* is sent offchain to the receiver

Transfer, receiver part

receiver_utxo **ACCEPT** with the *consignement* and the *outpoint blinding factor*