BREEZ SDK - NODELESS

# PLAN ₿ EDITION

# Prerequisites and Course Information

To be able to get the most out of this lecture, you'll need some prior knowledge on the following topics:

Bitcoin and Layer-2 Networks

Rust Programming Language

Go Programming Language

## COURSE DETAILS

**Lecturer:** Antonio (**hydra-yse**)

**Duration:** 3 hours

**Required Software:** None

### NOTICE ON THE RELIABILITY OF THE MATERIAL

The parts of the SDK which will be discussed today are still under active development, yet the inner logic of what will be shown has been consolidated, for the most part.

Therefore, expect this lecture to have a life expectancy of at least **1 year**, for most what will be presented.

# Lecture Overview

By the end of the lecture, you'll have:

1. Learned all the tools necessary to **contribute** to the Nodeless SDK, as well as some of our other implementations
2. Gained detailed knowledge on the **challenges** addressed by Bitcoin library developers and how we have tackled them
3. Developed the inspiration/insight to build your ***own*** Bitcoin/Lightning toolkits

To that extent, the lecture is divided into two parts:

## PART I

### Overview
General structure of the SDK, and which necessities are addressed by each part

### Primary Payment Logic
Deep-dive into the main payment flows: services, dependencies and use-cases

## PART II

### Secondary Services
Real-time synchronization & recovery, Side-Swap and SDK plugins

### On the Horizon
Overview of the SDK roadmaps, and what you can do to contribute

# Setup

If you'd like to follow along, you can clone the repository like so:

```bash
git clone https://github.com/breez/breez-sdk-liquid
cd breez-sdk-liquid
git checkout ead539d5dcd094efb95641eb01a7504d303421c9
```

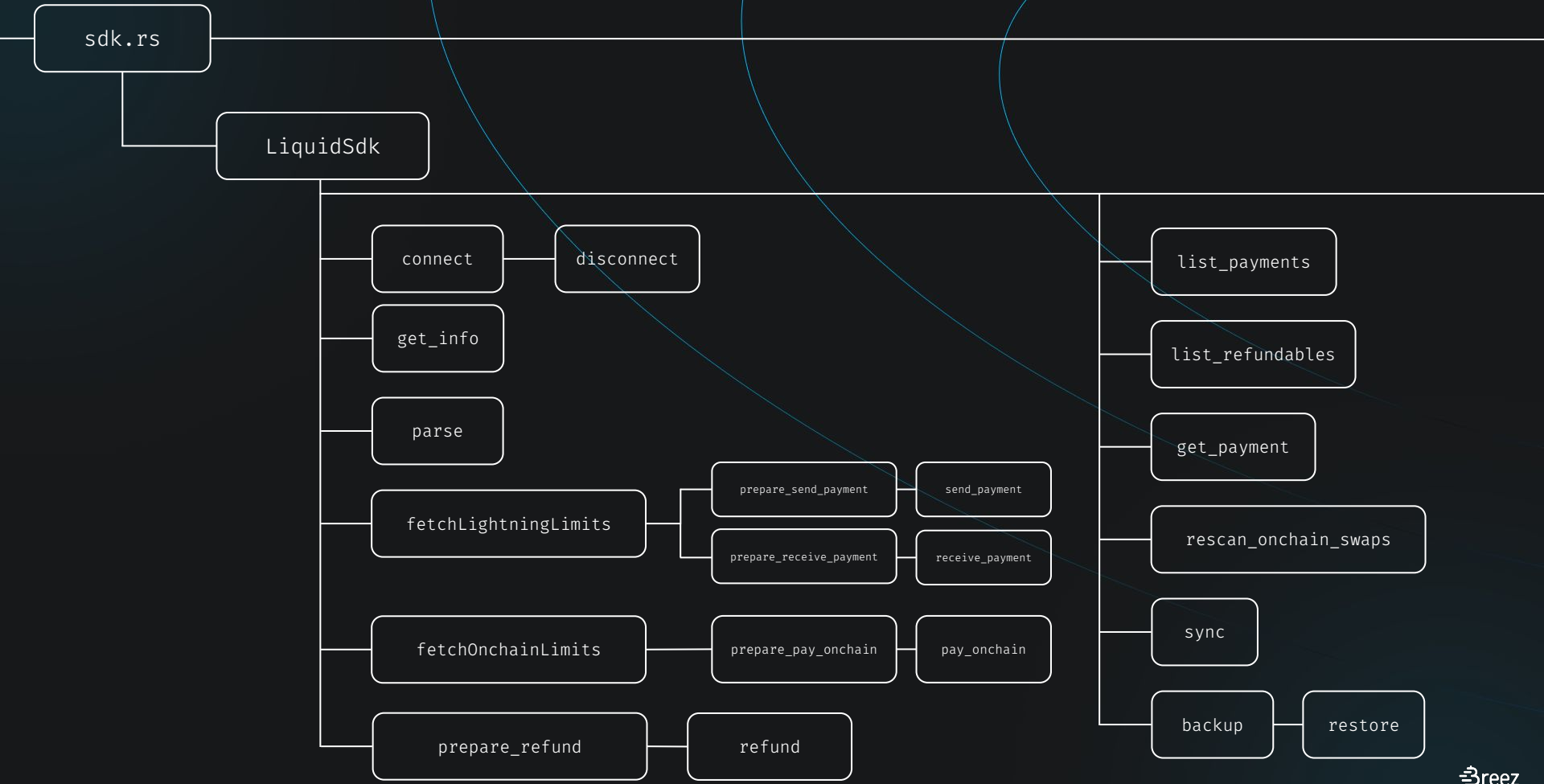
## Notice

For future readers, we will be checking out on commit **ead539d5dcd094efb95641eb01a7504d303421c9** which will be the reference point for the rest of the lecture

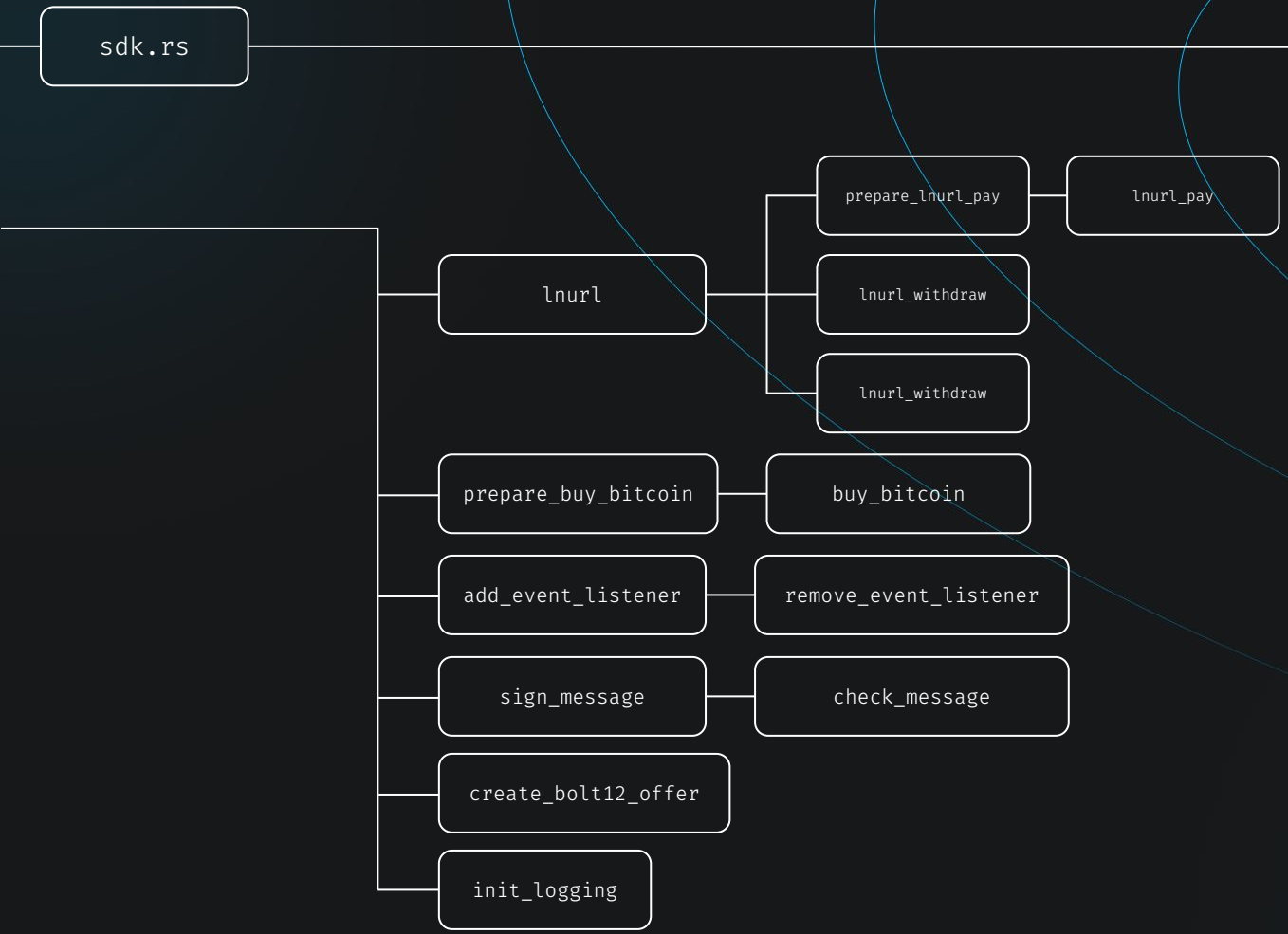
Please make sure you've also got an IDE and the rust toolchain setup (preferably including rust-analyzer). You can download it via your preferred package manager or via the following script (**rustup**):

```bash
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

# PART I


sdk.rs
LiquidSdk
connect
disconnect
get_info
parse
fetchLightningLimits
prepare_send_payment
send_payment
prepare_receive_payment
receive_payment
fetchOnchainLimits
prepare_pay_onchain
pay_onchain
prepare_refund
refund
list_payments
list_refundables
get_payment
rescan_onchain_swaps
sync
backup
restore

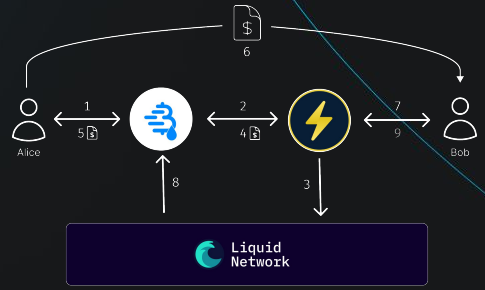
Breez


sdk.rs
lnurl
prepare_lnurl_pay
lnurl_pay
lnurl_withdraw
lnurl_withdraw
prepare_buy_bitcoin
buy_bitcoin
add_event_listener
remove_event_listener
sign_message
check_message
create_bolt12_offer
init_logging

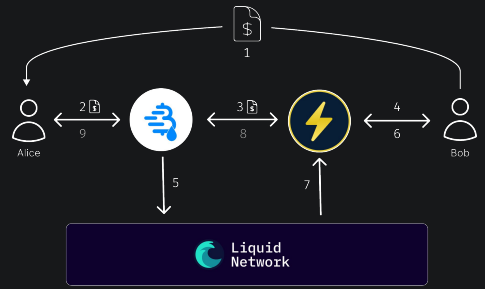
Breez

swapper

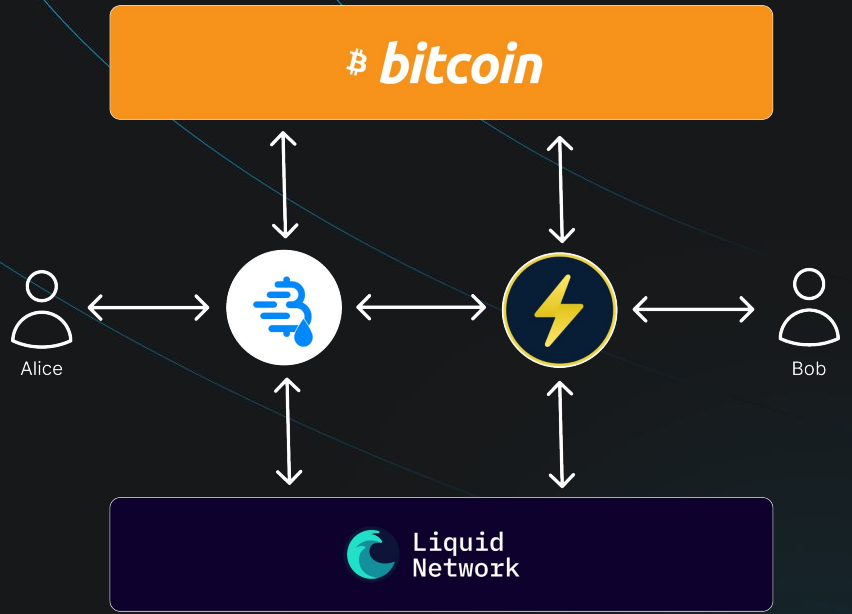
LN Receive Flow: Bob paying Alice (BTC to L-BTC)



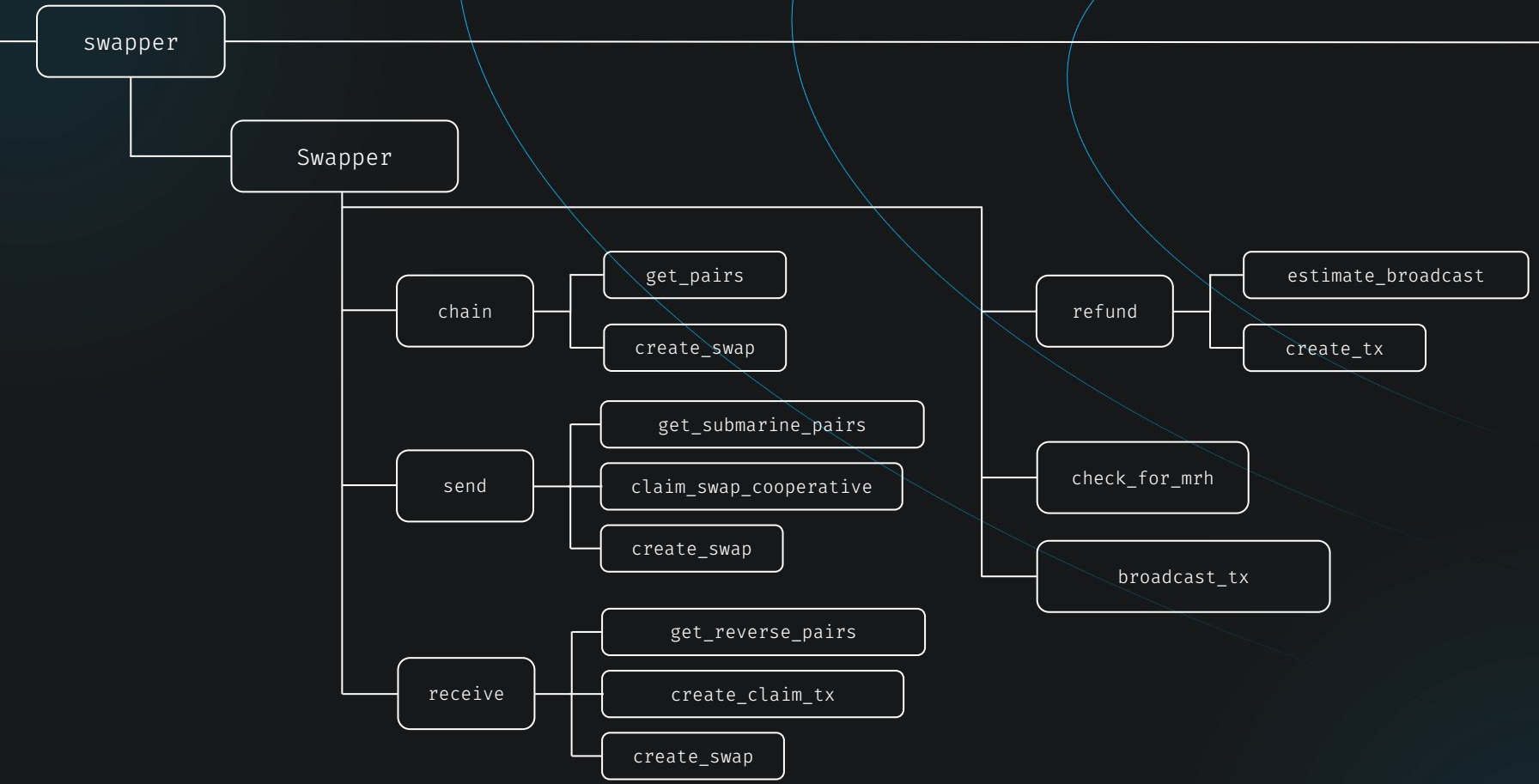

LN Send Flow: Alice paying Bob (L-BTC to BTC)

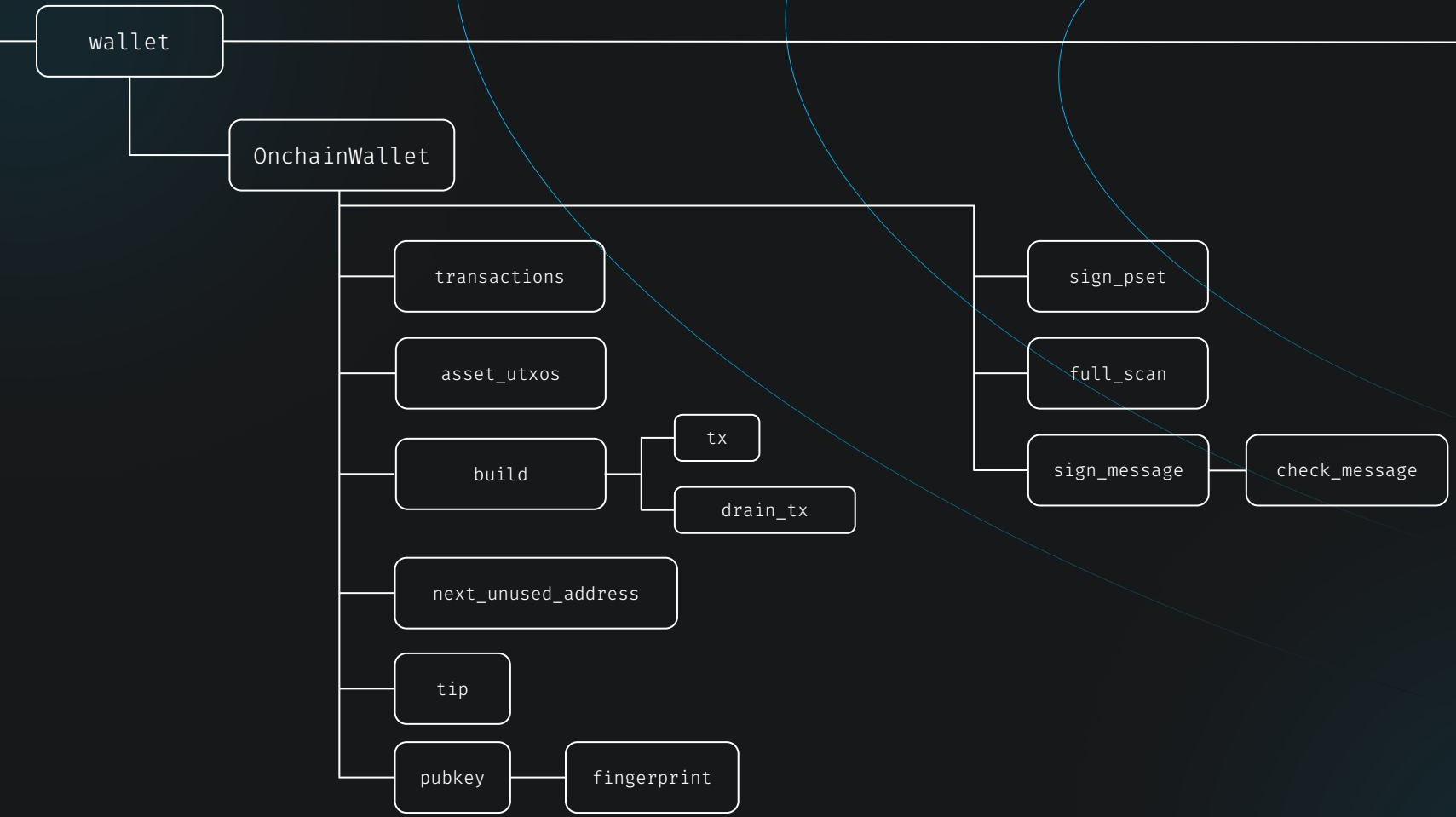



General BTC/L-BTC Flow (Chain Swaps)

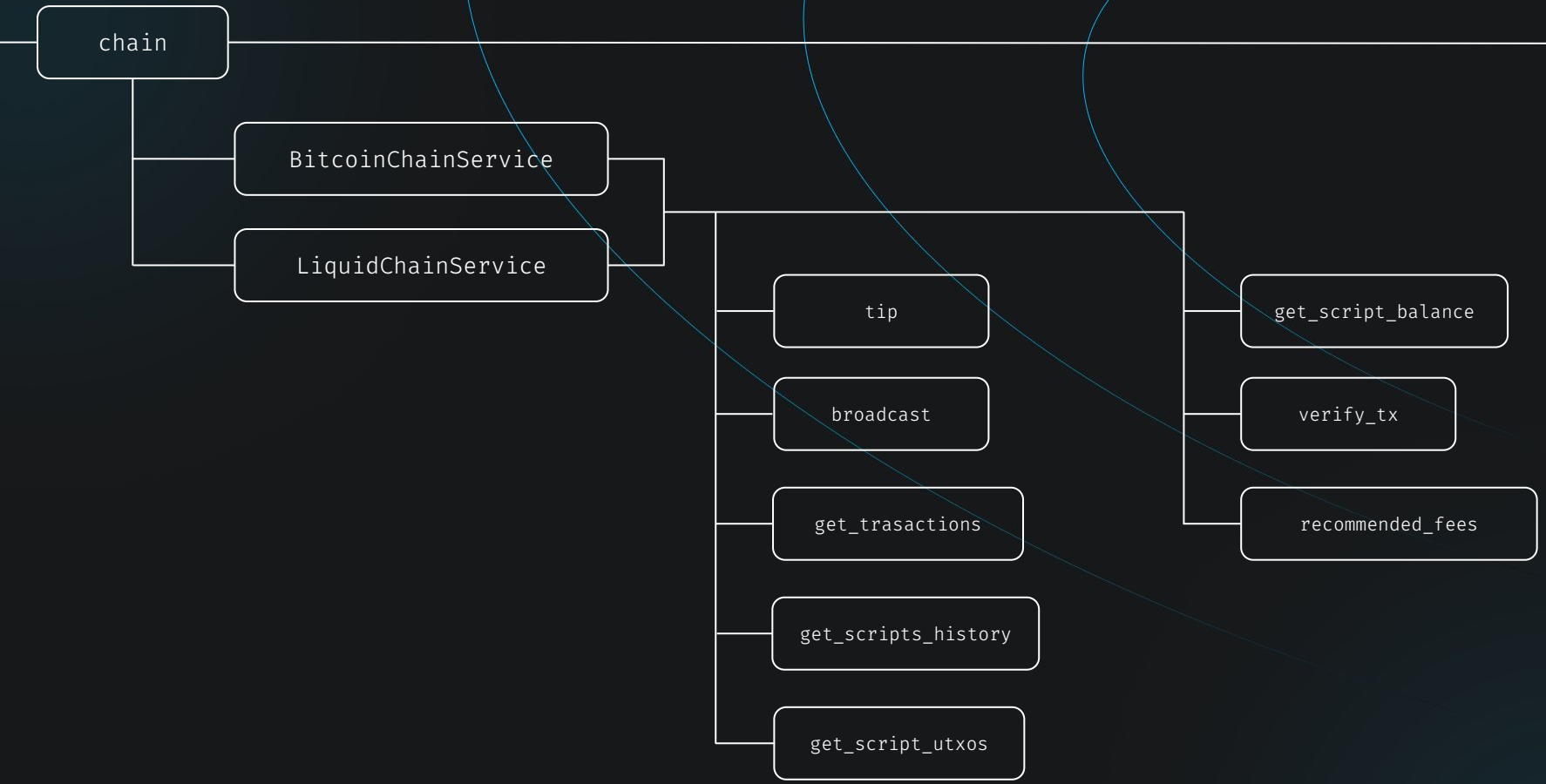



Breez


swapper
Swapper
chain
get_pairs
create_swap
send
get_submarine_pairs
claim_swap_cooperative
create_swap
receive
get_reverse_pairs
create_claim_tx
create_swap
refund
estimate_broadcast
create_tx
check_for_mrh
broadcast_tx


wallet
OnchainWallet
transactions
asset_utxos
build
tx
drain_tx
next_unused_address
tip
pubkey
fingerprint
sign_pset
full_scan
sign_message
check_message

Breez

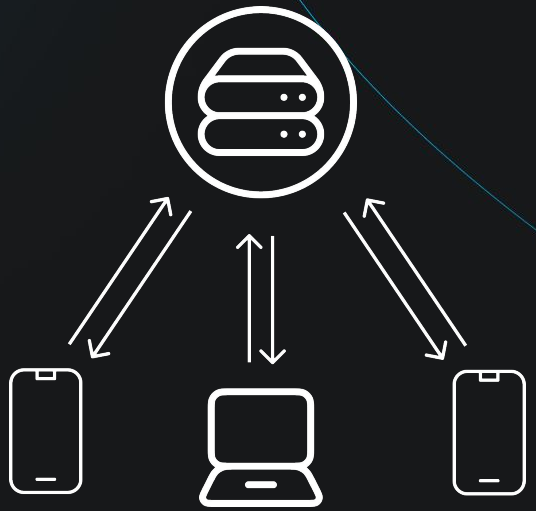

chain
BitcoinChainService
LiquidChainService
tip
broadcast
get_trasactions
get_scripts_history
get_script_utxos
get_script_balance
verify_tx
recommended_fees

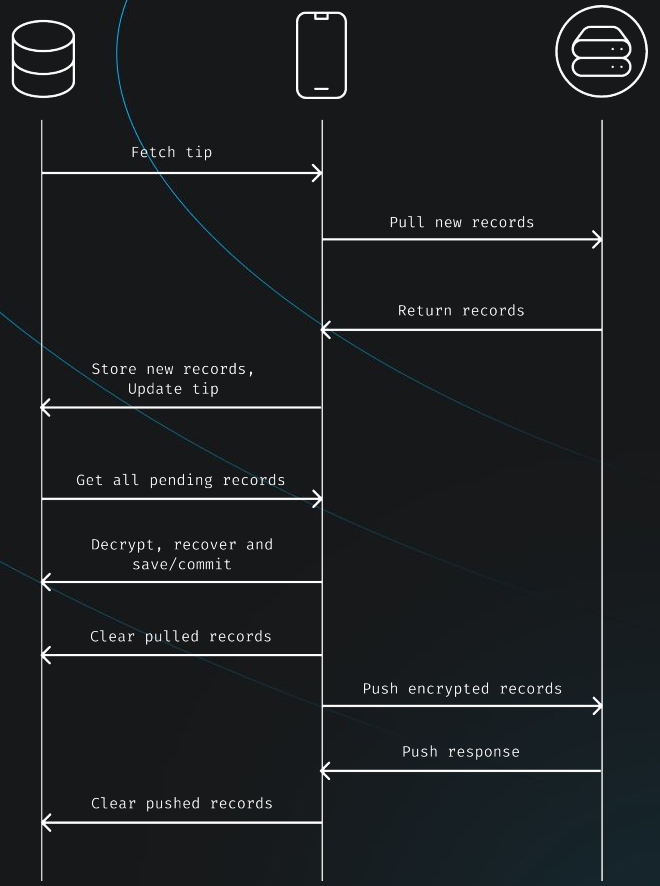

signer
Signer
xpub
sign_ecdsa
slip77_master_blinding_key
hmac_sha256
ecies
encrypt
decrypt

# END OF PART I!

# PART II

# Real-time Sync



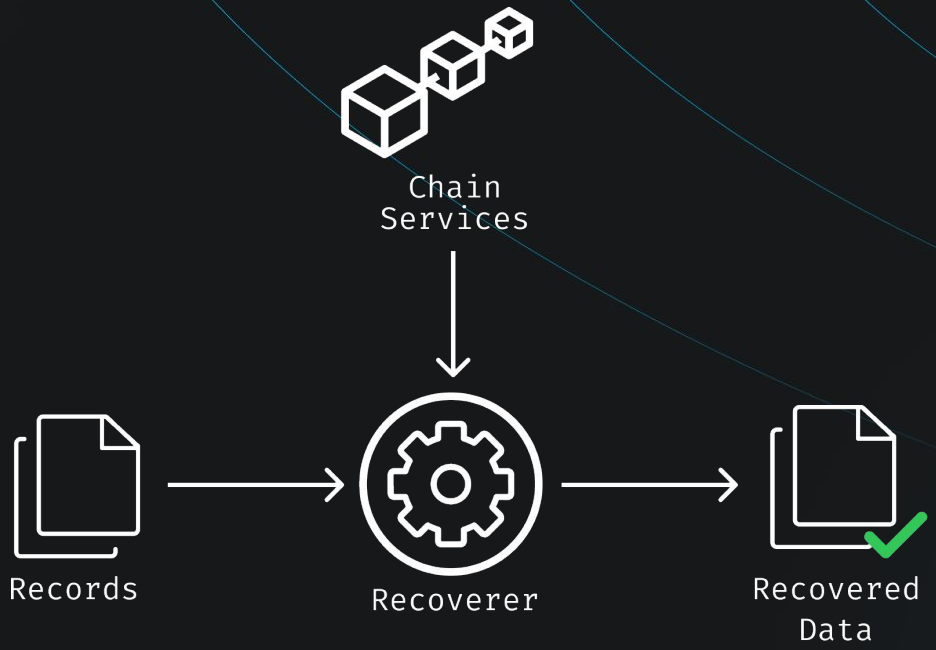



## Use-Case

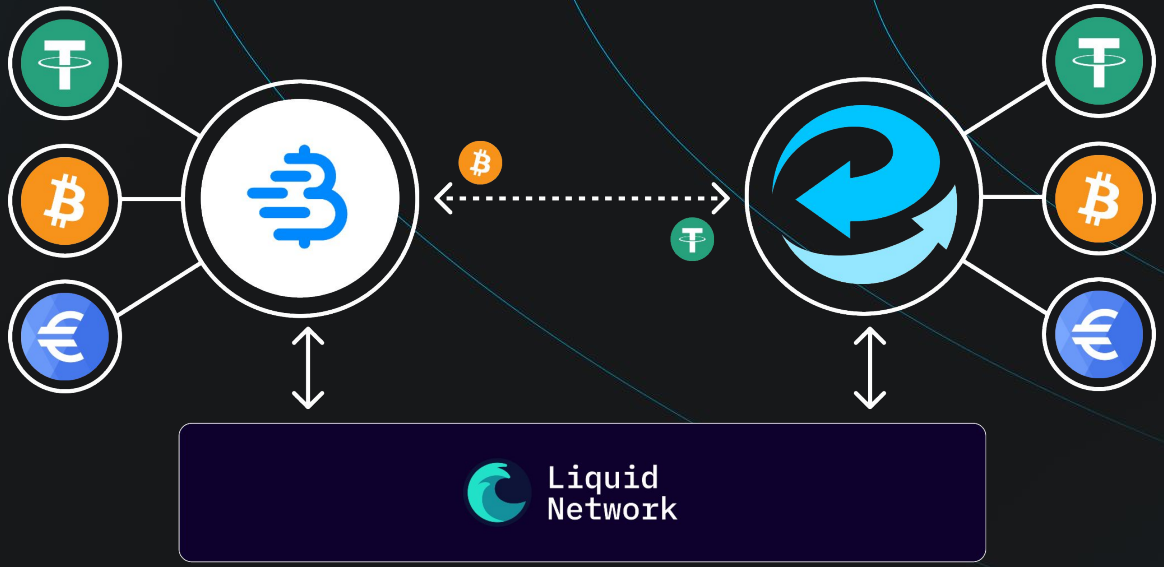
Synchronizing data across multiple devices in a secure, private and self-hostable manner.
You can find the repository for the server implementation **here**.

# Recovery

The real-time sync is optimized by only saving on the remote data that is strictly non-derivable (from onchain data). Once we pull new the necessary information from the service, we now have to **cross-reference** it with the blockchain. This process is called **recovery**.



# Side-Swap




## Use-Case

Enabling **multi-asset** transactions on the SDK, via the Side-Swap API.
The SDK listens to market changes in real time and settles on a rate, then the **trade** begins.
If you want to pay someone with an asset you don't own, the exchange can be executed **in-flight**.

Coming soon!

# Plugins and NWC

## Use-Case

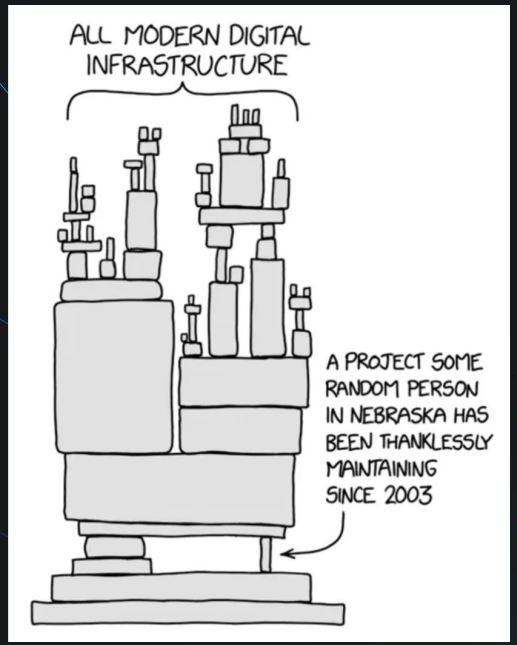
Enabling **user-defined** plugins that can directly interact with internal SDK tools, such as storage, as well as defining an SDK-bound object with a defined lifetime.
As of now, the interface just exposes a restricted set of features.

# Roadmap & Contributing

We are constantly working with new people and projects, so go ahead and reach out!
Let's make Bitcoin and open-source great **together**!

**SDK Development Roadmap**

- [x] Send/Receive Lightning payments
- [x] CLI Interface
- [x] Foreign languages bindings
- [x] Export/Import SDK data
- [x] Pay BTC on-chain
- [x] Receive via on-chain address
- [x] LNURL-Pay
- [x] LNURL-Withdraw
- [x] Send to a Lightning address
- [x] Receive via Lightning address
- [x] Webhook for receiving payments
- [x] Offline receive via notifications
- [x] Offline swaps via notifications
- [x] Real-time sync
- [x] External input parsers
- [x] Bolt12 send
- [x] BIP353 pay codes
- [x] Amountless BTC swaps
- [x] Support USDT and other Liquid assets
- [x] Pay fees with USDT
- [x] Lower minimum payment amount
- [x] WebAssembly
- [x] Bolt12 receive
- [x] Add fees via a dedicated portal
- [x] USDT <-> LBTC swaps
- [ ] WebLN
- [ ] NWC




Contributions are ***always*** welcome!

# THANKS FOR WATCHING

SEE YOU IN THE NEXT ONE...