LNP/BP Standards Association

RGB Wallets

integration perspective

LNP/BP Standards Association

Developed by **Dr Maxim Orlovsky**

Sponsored by **Pandora Core AG**

RGB & LNP today

- >**1.5 years** of work on the current RGB version
  (code & specs started from scratch in mid-2019)
- >**30** pure **rust libraries** (crates) with >**0.5 million lines** of Rust code
- >**100 contributions** (issues, PRs),
  tens of thousands lines of code for Rust Bitcoin ecosystem
- >**20 contributors** to issues, docs & code
- **2 wallets** actively being developed (Bitfinex, Pandora Core “MyCitadel”)
- **1 exchange** working on integration + **2** others considering to join
- **~2 known projects** outside of LNP/BP Association integrating RGB

Current status & priorities

- v0.3 released last week
- v0.4 got into development
  - Completion of SDKs and first wallet(s)
  - Full LN compatibility with other LN nodes on bitcoin transactions
  - Universal invoices
  - Better & safer PSBT support

Main actors

- Users: explain human user interaction with UI
- Wallet components: UI, controller (business logic), storage (data requiring backup & management)
- HSM: devices or software used for signing transactions
- Network: bitcoin P2P network, LN, electrum services; Bifrost services in future

consulting
white-label solutions
Pandora Core
LNP/BP Association
contributing
People
Done by wallet devs
Third-parties
Public good
User
Wallet UI
Wallet Controller
Wallet Storage
HSM
RGB SDK / Node
Network

1. Creating invoice

User
Wallet UI
Wallet Controller
Wallet Storage
RGB SDK / Node
Invoice creation
asks for invoice
create_invoice
invoice
(invoice, blinging_secret)
store(blinding_secret)
(invoice)
presents invoice
bech32 string or QR
User
Wallet UI
Wallet Controller
Wallet Storage
RGB SDK / Node

Universal invoices

Supporting Bitcoin, LN, RGB and other types of assets

LNPBP-38 standard & implementation progress report

What is it?

- Presented & discussed in previous calls
- Bech32-encoded strings supporting
  - Both on-chain & lightning
  - Multiple asset formats (including Liquid, RGB and potentially others)
  - Payment aggregation
  - Hierarchical wallet receivers for on-chain payments
  - Good extensibility with TLVs

Updates

- No need in network id: AssetId plays the role
- Getting rid of payees field
- Using PSBT beneficiary for paying multiple parties at the same time
  - price field is indicative for the total amount to pay; must be split in equal amounts between 0-outputs of PSBT
- Re-using lock field for both HTLCs and PTLCs (in Schnorr key format). PTLC is indicated with a feature field
- Using details field to specify connectivity with merchant for getting payment slip
- TLV extensions will be used if needed for RGB-21, 22, 23 invoices
  - “send me NFT”
  - “send me your identity”

Materials

- Discussion: https://github.com/LNP-BP/LNPBPs/discussions/82

- Initial implementation: https://github.com/LNP-BP/rust-lnpbp/blob/master/invoice/src/invoice.rs

Bech32

Its use in RGB & LNP ecosystem

LNPBP-39 standard & implementation progress report

Non-RGB specific

- **data1...**
  - encoding large chunks of binary data
  - transfer between apps via messengers, e-mail, copy/paste
  - simplifies debugging
- **z1...**
  - uses standard compression algorithm
  - allows versioning of compression
  - the same as above, but uses shorter strings for compressible data
- **id1...**
  - Taproot-style tagged hashes of data
  - used for simplifying different forms of ids
- **i1...**
  - Universal invoices described above

RGB-specific

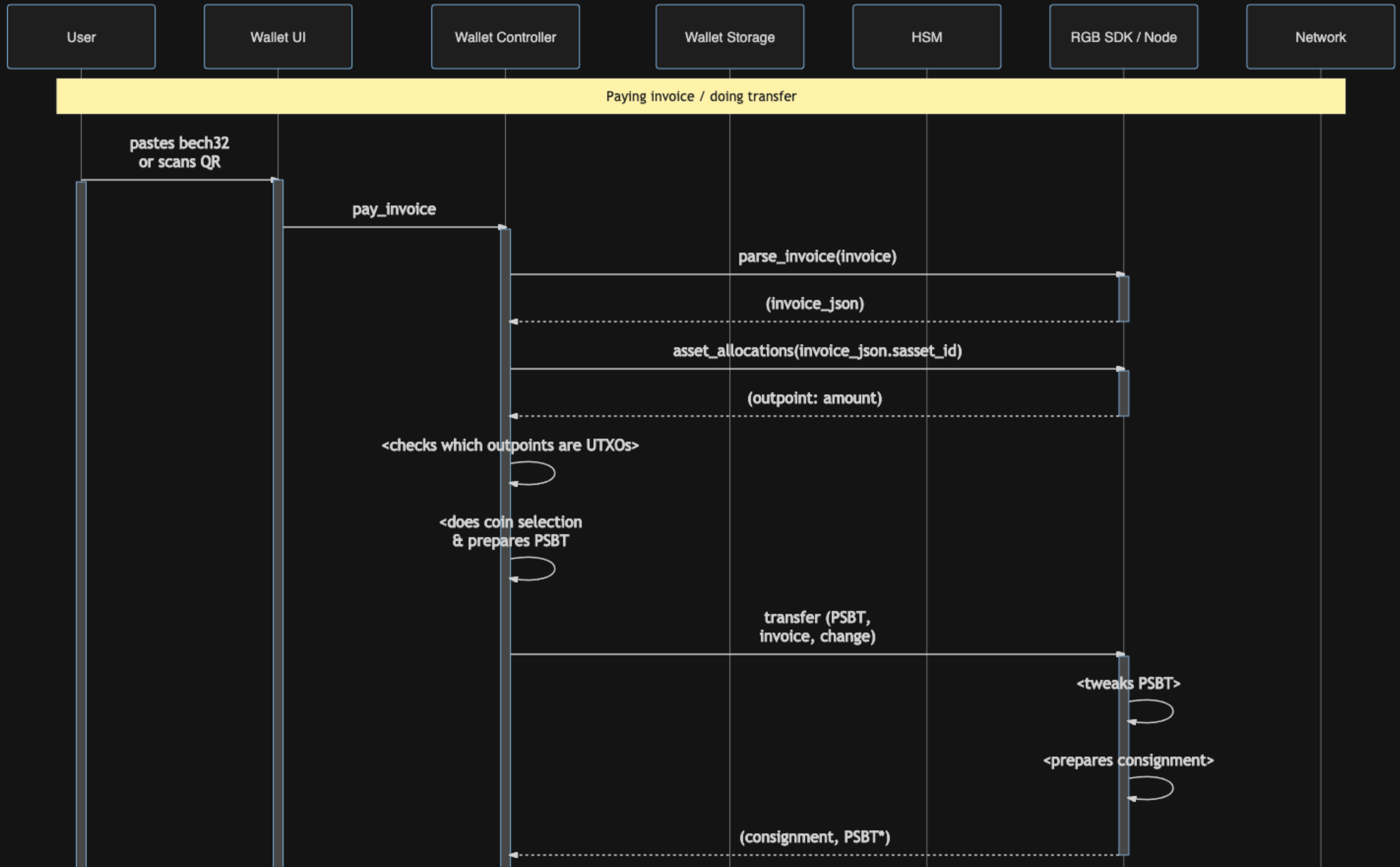
- **rgb1...** – contract id (genesis id)
  unique asset identifier
- **genesis1...** - genesis data
  distributes asset
- **consignment1...** - consignment data
  sends assets to the receiver (compressed format)
- **utxob1...** - blinded UTXO data
  part of the older invoices; may be used for specifying payment destination with RGB
- **sch1...** — schema id
  specifying asset type (fungible, NFT…)
- **schema1...** - schema data
  distributing custom schemata

Materials

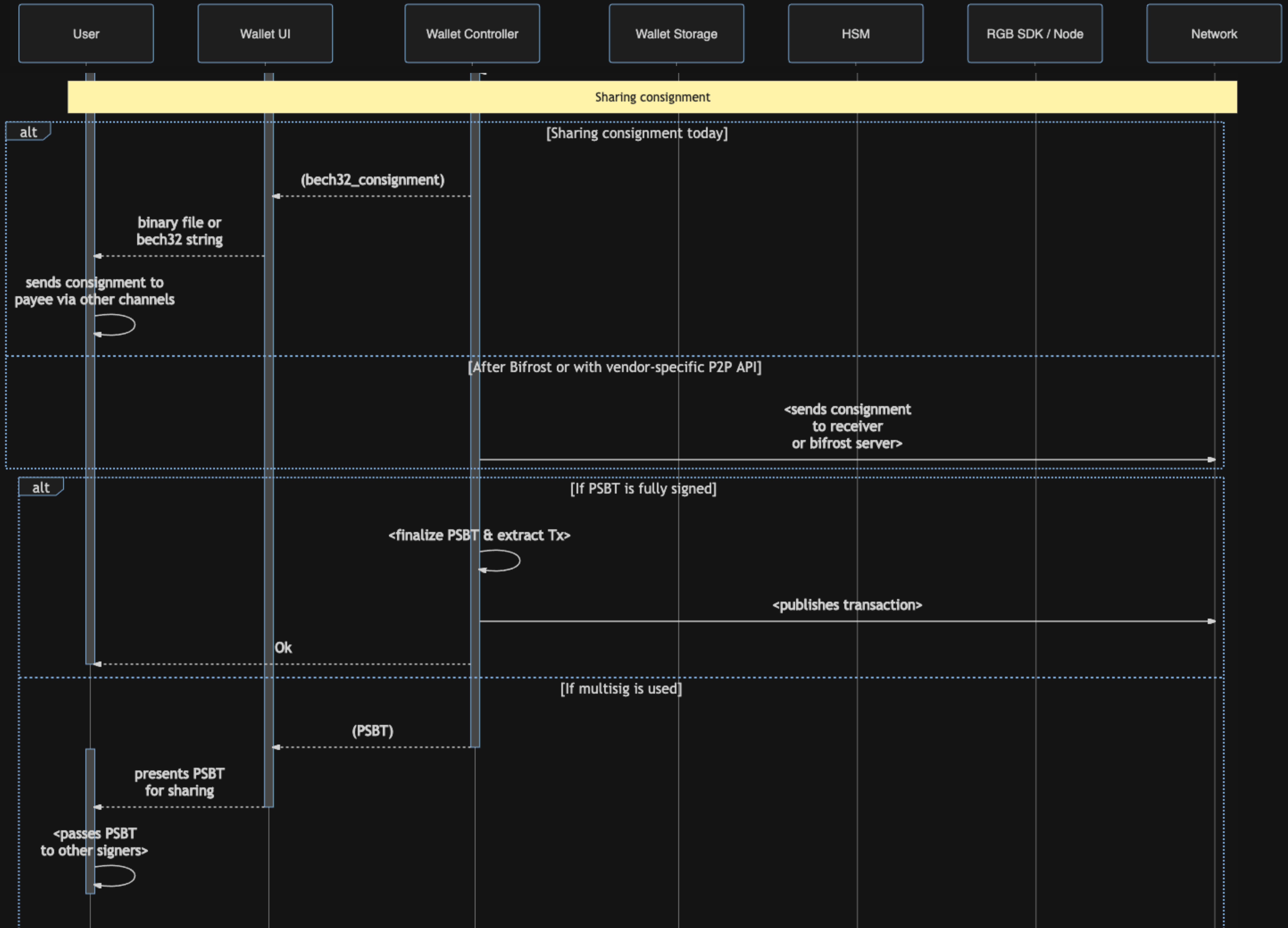
- Discussion: https://github.com/LNP-BP/LNPBPs/discussions/85

- Initial implementation:
    - Common: https://github.com/LNP-BP/rust-lnpbp/blob/master/src/bech32.rs
    - RGB: https://github.com/rgb-org/rgb-core/blob/master/src/bech32.rs

2. Paying invoice


User
Wallet UI
Wallet Controller
Wallet Storage
HSM
RGB SDK / Node
Network
Paying invoice / doing transfer
pastes bech32 or scans QR
pay_invoice
parse_invoice(invoice)
(invoice_json)
asset_allocations(invoice_json.sasset_id)
(outpoint: amount)
<checks which outpoints are UTXOs>
<does coin selection & prepares PSBT
transfer (PSBT, invoice, change)
<tweaks PSBT>
<prepares consignment>
(consignment, PSBT*)

User
Wallet UI
Wallet Controller
Wallet Storage
HSM
RGB SDK / Node
Network
preparing PSBT
process_tweaks(PSBT)
<extracts output tweaks, fills in known input tweaks>
(PSBT* with tweaks)
sign_psbt
<checks PSBT data>
<signs>
(signed_psbt)


User
Wallet UI
Wallet Controller
Wallet Storage
HSM
RGB SDK / Node
Network
Sharing consignment
alt
[Sharing consignment today]
(bech32_consignment)
binary file or bech32 string
sends consignment to payee via other channels
[After Bifrost or with vendor-specific P2P API]
<sends consignment to receiver or bifrost server>
alt
[If PSBT is fully signed]
<finalize PSBT & extract Tx>
<publishes transaction>
Ok
[If multisig is used]
(PSBT)
presents PSBT for sharing
<passes PSBT to other signers>

User
Wallet UI
Wallet Controller
Wallet Storage
HSM
RGB SDK / Node
Network
Variant for multisig wallets
opt
[Other signers]
provides PSBT
sign(PSBT)
process_tweaks(PSBT)
<extracts output tweaks, fills in known input tweaks>
(PSBT* with tweaks)
sign_psbt
<checks PSBT data>
<signs>
(signed_psbt)
<stores PSBT>

User
Wallet UI
Wallet Controller
Wallet Storage
HSM
RGB SDK / Node
Network
alt
[If PSBT is fully signed]
<finalize PSBT & extract Tx>
<publishes transaction>
Ok
[If multisig is used]
(PSBT)
presents PSBT for sharing
<passes PSBT to other signers>

Partially signed bitcoin transactions

RGB support & hardware wallets interoperability

Partially signed bitcoin transaction

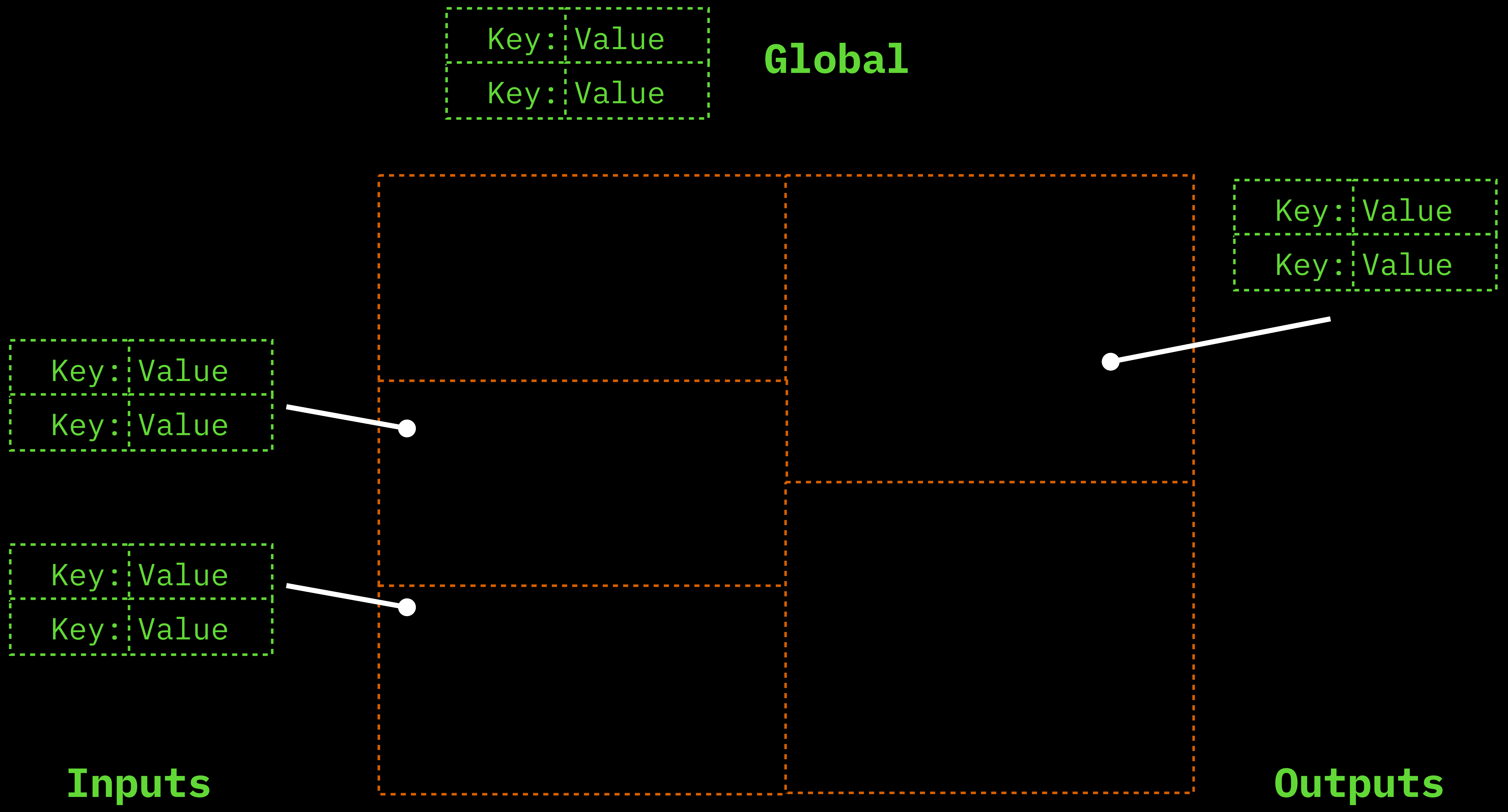
Attaches key-value maps with extended information to bitcoin transaction

- Global:
  - list of xpubs
  - hashes
- Inputs
  - previous transactions (needed to compute fee etc)
  - signatures
- Outputs
  - full script sources
  - key derivation information

Partially signed bitcoin transaction

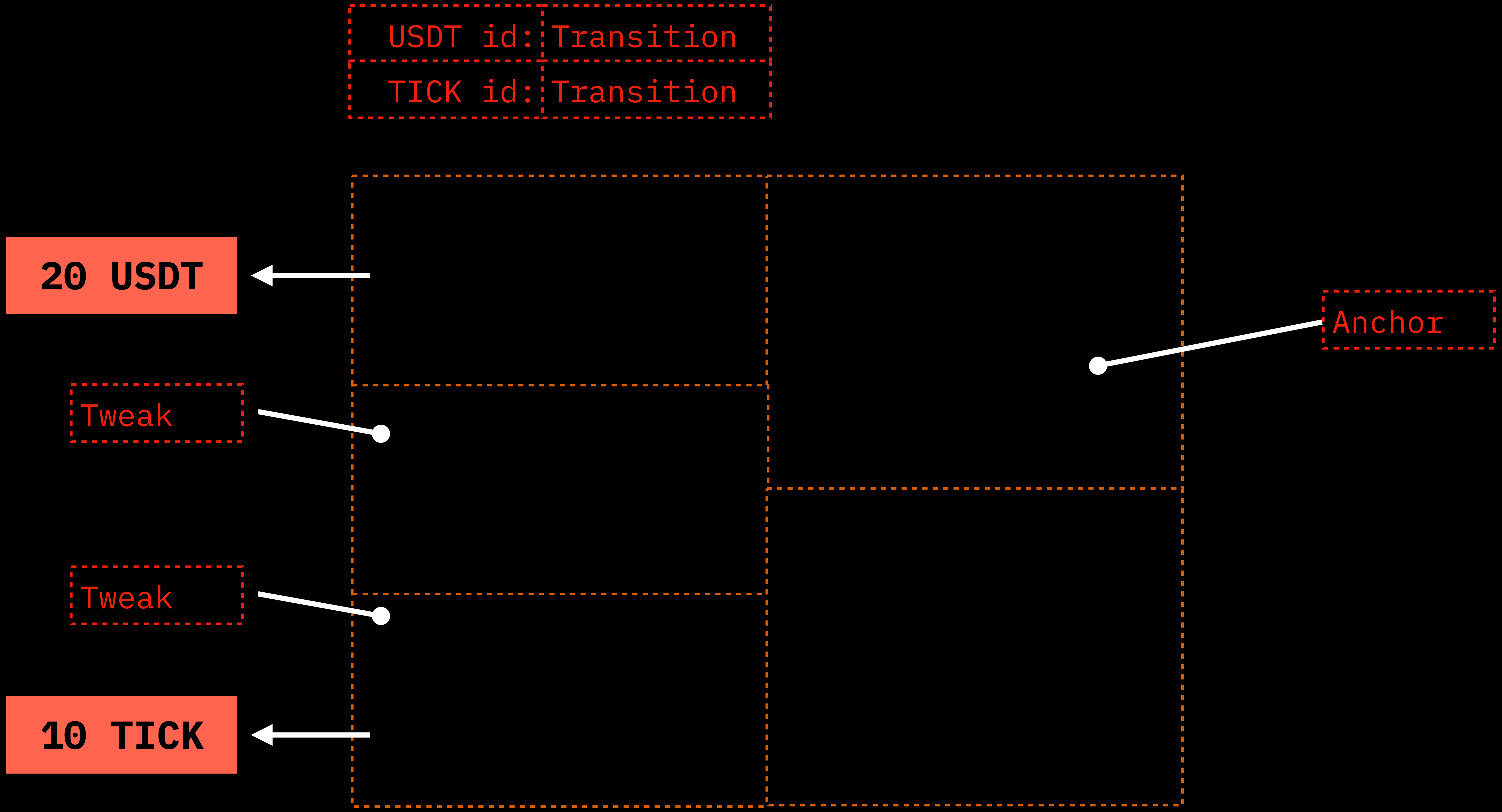
Attaches key-value maps with extended information to bitcoin transaction

- Global:
  - list of xpubs
  - hashes
- Inputs
  - previous transactions (needed to compute fee etc)
  - signatures
- Outputs
  - full script sources
  - key derivation information



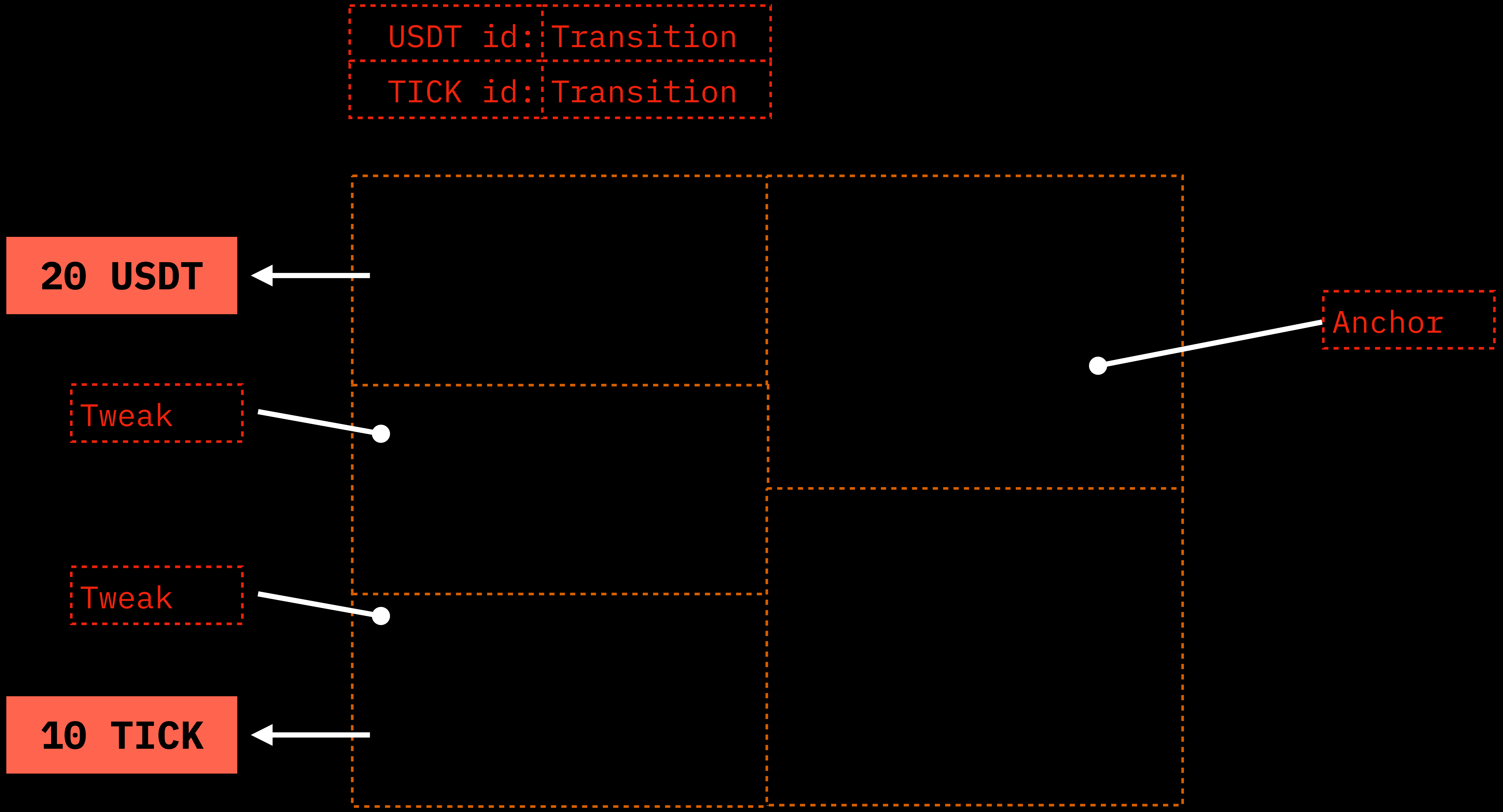
Adding RGB to PSBT

- Inputs:
  - public key tweaks required to create valid signatures
  - one per input
  - may apply to inputs w/o RGB assets
- Outputs:
  - single anchor per output
  - multiple outputs can contain actors
  - used to verify that commitment does not lead to funds loss
- Global:
  - Source of the state transitions committing to the transaction; required to verify commitment



What we can check

- Assets are spent with proper commitment
- We are not embedding any other tweak
- Amounts we are transferring with the transaction
- With this PSBT we keep the required tweak data for future spending
  (tweaks for spending PSBT are copied from anchor data of the PSBT spent)

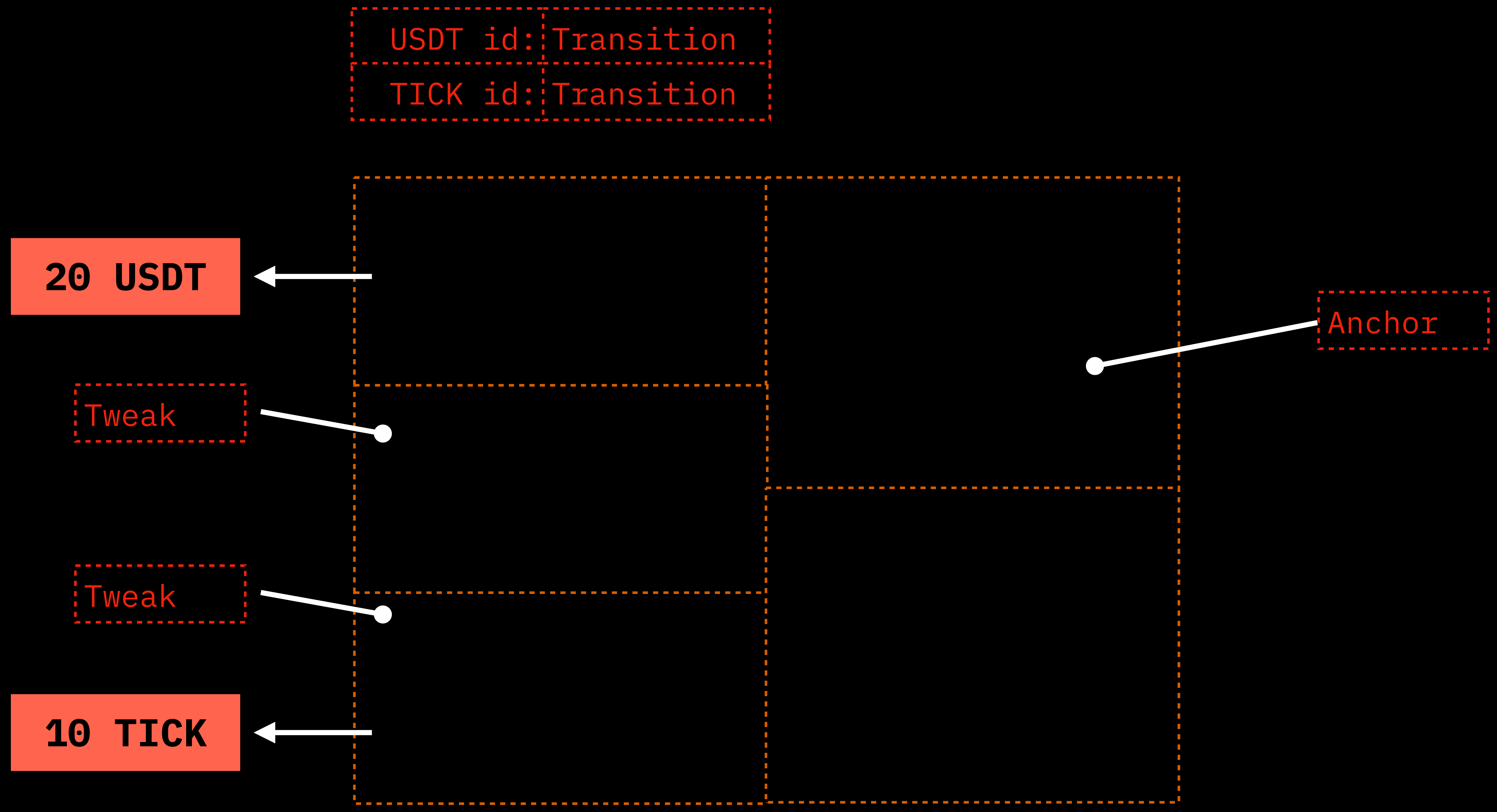


What we can’t check

- If there are other assets assigned to the same outputs. They will be lost

- Amounts of assets spent is real (can be checked with UI)

- The spending is a valid spending not leading to the loss of all assets



Action points

- Propose BIP-174 PR for tweaks in input keys
  (the rest must be kept vendor-specific)
- Implement as a part of RGB Core library & add support in RGB Node
  (scheduled for v0.4 in Feb)
- Work with hardware wallet developers to do software for working
  with RGB-enabled PSBTs

3. Receiving payment

- Receiving consignment
- Validating consignment
- Accepting consignment (adding it to stash)

User
Wallet UI
Wallet Controller
Wallet Storage
RGB SDK / Node
Electum, Bifrost, Bitcoin Core
Receiving consignment
alt
[Today]
Receives consignment via other channels
imports consignment data
process(consignment)
[With vendor-specific P2P API]
(consignment)
[After Bifrost]
<checks with bifrost server>
(consignment)

User
Wallet UI
Wallet Controller
Wallet Storage
RGB SDK / Node
Electum,
Bifrost,
Bitcoin Core
Verifying consignment
verify(consignment)
<talks to
Electrum server>
<validate graph, schema,
execute scripts>
(validation_report)
Be aware that
witness tx may
not be mined yet
reported as failure
opt
[If there were warnings or
validation errors]
([errors])
informs user
Wait for witness tx
to be mined, if needed

User
Wallet UI
Wallet Controller
Wallet Storage
RGB SDK / Node
Electum, Bifrost, Bitcoin Core
Accepting consignment
extract_outpoints(consignment)
([blind_utxos])
find_secrets([blind_utxos])
([blinding_secret])
accept(consignment, [blinding_secret])
<stores data in stash updates assets cache>
Ok
sync
(allocation_info)
<Update UI>
inform about received funds