

This spec captures the actions and states of bitcoin transactions in the context of the bitcoin blockchain. These actions will be used by the *LN Contracts* spec and other layer two contract specifications.

The focus of this module is to provide:

1. Way to generate transactions that accept input and generate outputs
2. Confirm transactions so that outputs can be spent.
3. Most importantly - provide a way to verify spend conditions without building the entire cryptography machinery. This enables spec authors to focus on what the conditions achieve instead of how those conditions are achieved.

Goal A: Move environment / bitcoin transaction actions and variables from *Contracts* to here

EXTENDS *Sequences*,
Integers,
TLC,
SequencesExt,
FiniteSetsExt

Define constants so that we can define finite sets for inputs, outputs and txids etc.

CONSTANTS	<i>CSV</i> ,	Set of <i>CSV</i> values
	<i>VOUT</i> ,	Set of vout values
	<i>TXID</i> ,	Set of transaction ids
	<i>AMOUNT</i> ,	Set of amounts that can be used
	<i>PARTY</i> ,	Parties participating in the <i>L2</i> protocol
	<i>KEY</i> ,	Set of keys for each party used
		in the <i>L2</i> protocol
	<i>HASH</i>	Set of all hash preimages

SighashFlag \triangleq {"all", "none", "single", "anyonecanpay"}

Set of output types supported for building contracts.

Each output type will have to provide a means to verify an input trying to spend it.

OutputTypes \triangleq {"p2wkh", "multisig", "multisig-with-csv", "hash-lock"}

OutputTypes \triangleq {"p2wkh", "multisig", "multisig-with-csv"}

NoCSV \triangleq CHOOSE $c : c \notin CSV$

MaxCSV \triangleq CHOOSE $c \in CSV : \forall y \in CSV : c \geq y$

NoHash \triangleq CHOOSE $h : h \notin HASH$

NoTxid \triangleq -1

NoSpendHeight \triangleq -1

All keys available for use by the parties
Keys $\triangleq PARTY \times KEY$

Input \triangleq [

$txid : TXID,$
 $index : VOUT,$
 $sighash_flag : SighashFlag,$ Parts of transactions covered by signature
 $signed_by : Seq(Keys),$ One or more keys that have signed this input
 $hash_preimage : HASH \cup \{NoHash\}$
]

$Output \triangleq [$
 $index : VOUT,$
 $type : OutputTypes,$
 $keys : Seq(Keys),$ Sig from these keys is required to spend
 $csv : CSV \cup \{NoCSV\},$ The *CSV* should have expired before spend
 $hash : HASH \cup \{NoHash\},$ Pre-image required to spend
 $amount : AMOUNT$
]

VARIABLES

$chain_height,$
 $transactions,$
 $mempool,$
 $published$

$CreateP2WKHOutput(keys, amount) \triangleq [$
 $index \mapsto 1,$
 $type \mapsto \text{"p2wkh"},$
 $keys \mapsto keys,$
 $csv \mapsto NoCSV,$
 $hash \mapsto NoHash,$
 $amount \mapsto amount$
]

$CreateMultisigOutput(keys, amount) \triangleq [$
 $index \mapsto 1,$
 $type \mapsto \text{"multisig"},$
 $keys \mapsto keys,$
 $csv \mapsto NoCSV,$
 $hash \mapsto NoHash,$
 $amount \mapsto amount$
]

$CreateMultisigWithCSVOutput(keys, amount) \triangleq [$
 $index \mapsto 1,$
 $type \mapsto \text{"multisig_with_csv"},$
 $keys \mapsto keys,$

```

    csv  $\mapsto$  MaxCSV,
    hash  $\mapsto$  NoHash,
    amount  $\mapsto$  amount
]

```

Create a transaction spending the given output/*id*, and spendable by the given key.

```

CreateP2WKHTx(spending_output, output_key, amount)  $\triangleq$  [
  inputs  $\mapsto$   $\langle$  [txid  $\mapsto$  spending_output[1],
    index  $\mapsto$  spending_output[2],
    sighash_flag  $\mapsto$  "all",
    signed_by  $\mapsto$  transactions[spending_output[1]].outputs[spending_output[2]].keys,
    hash_preimage  $\mapsto$  NoHash],
  outputs  $\mapsto$   $\langle$  CreateP2WKHOutput(output_key, amount)  $\rangle$ 
]

```

Create a transaction spending the given output/*id*, and spendable by as a *multisig* of the given keys.

```

CreateMultisigTx(spending_output, output_keys, amount)  $\triangleq$  [
  inputs  $\mapsto$   $\langle$  [txid  $\mapsto$  spending_output[1],
    index  $\mapsto$  spending_output[2],
    sighash_flag  $\mapsto$  "all",
    signed_by  $\mapsto$  transactions[spending_output[1]].outputs[spending_output[2]].keys,
    hash_preimage  $\mapsto$  NoHash],
  outputs  $\mapsto$   $\langle$  CreateMultisigOutput(output_keys, amount)  $\rangle$ 
]

```

```

CreateUnsingedMultisigTx(spending_output, output_keys, amount)  $\triangleq$  [
  inputs  $\mapsto$   $\langle$  [txid  $\mapsto$  spending_output[1],
    index  $\mapsto$  spending_output[2],
    sighash_flag  $\mapsto$  "all",
    signed_by  $\mapsto$   $\langle$   $\rangle$ ,
    hash_preimage  $\mapsto$  NoHash],
  outputs  $\mapsto$   $\langle$  CreateMultisigOutput(output_keys, amount)  $\rangle$ 
]

```

```

CreateMultisigWithCSVtx(spending_output, output_keys, amount)  $\triangleq$  [
  inputs  $\mapsto$   $\langle$  [txid  $\mapsto$  spending_output[1],
    index  $\mapsto$  spending_output[2],
    sighash_flag  $\mapsto$  "all",
    signed_by  $\mapsto$  transactions[spending_output[1]].outputs[spending_output[2]].keys,
    hash_preimage  $\mapsto$  NoHash],
  outputs  $\mapsto$   $\langle$  CreateMultisigWithCSVOutput(output_keys, amount)  $\rangle$ 
]

```

Choose keys to use in outputs. It is used by *AddSpendTxToMempool*.

We expect both this expression and the *AddSpendTxToMempool* action to be provided by the layer 2 protocol spec.

$$\begin{aligned} \text{ChooseOutputKeys}(\text{output_type}) &\triangleq \\ \text{IF } \text{output_type} = \text{"p2wkh"} & \\ \text{THEN } \text{SetToSeq}(\text{CHOOSE } k \in k\text{Subset}(1, \text{Keys}) : \text{TRUE}) & \\ \text{ELSE } \text{SetToSeq}(\text{CHOOSE } k \in k\text{Subset}(2, \text{Keys}) : \text{TRUE}) & \end{aligned}$$

Return TRUE if the given output uses a key for the provided party

$$\begin{aligned} \text{OutputOwnedByParty}(o, p) &\triangleq \\ p \in \{k[1] : k \in \text{ToSet}(\text{transactions}[o[1]].\text{outputs}[o[2]].\text{keys})\} & \end{aligned}$$

$$\begin{aligned} \text{ConfirmedTransactions} &\triangleq \\ \{p \in \text{DOMAIN } \text{transactions} : \text{published}[p] \neq \text{NoSpendHeight}\} & \end{aligned}$$

$$\begin{aligned} \text{AllOutputs} &\triangleq \\ \text{UNION } \{\{txid\} \times \{o.\text{index} : o \in \text{ToSet}(\text{transactions}[txid].\text{outputs})\} : txid \in \text{ConfirmedTransactions}\} & \end{aligned}$$

$$\begin{aligned} \text{SpentOutputs} &\triangleq \\ \{\langle i.txid, i.index \rangle : i \in & \\ \text{UNION } \{\text{ToSet}(\text{transactions}[txid].\text{inputs}) : txid \in \text{ConfirmedTransactions}\} & \\ \} & \end{aligned}$$

$$\text{UnspentOutputs} \triangleq \text{AllOutputs} \setminus \text{SpentOutputs}$$

Add a coinbase *tx* spendable with a pk. No verification is required here as no prevout is being spent.

$$\begin{aligned} \text{AddP2WKHCoinbaseToMempool}(id, keys, amount) &\triangleq \\ \wedge id \notin \text{mempool} & \\ \wedge \text{published}[id] = \text{NoSpendHeight} & \\ \wedge \text{transactions}' = [\text{transactions} \text{ EXCEPT } ![id] = [\text{inputs} \mapsto \langle \rangle, & \\ \text{outputs} \mapsto \langle \text{CreateP2WKHOutput}(keys, amount) \rangle]] & \\ \wedge \text{mempool}' = \text{mempool} \cup \{id\} & \\ \wedge \text{UNCHANGED } \langle \text{chain_height}, \text{published} \rangle & \end{aligned}$$

Add a coinbase *tx* with a *multisig* output spendable by signature from all keys.

We don't do threshold signatures for simplicity.

$$\begin{aligned} \text{AddMultisigCoinbaseToMempool}(id, keys, amount) &\triangleq \\ \wedge id \notin \text{mempool} & \\ \wedge \text{published}[id] = \text{NoSpendHeight} & \\ \wedge \text{transactions}' = [\text{transactions} \text{ EXCEPT } ![id] = [\text{inputs} \mapsto \langle \rangle, & \\ \text{outputs} \mapsto \langle \text{CreateMultisigOutput}(keys, amount) \rangle]] & \\ \wedge \text{mempool}' = \text{mempool} \cup \{id\} & \\ \wedge \text{UNCHANGED } \langle \text{chain_height}, \text{published} \rangle & \end{aligned}$$

Confirm transaction from *mempool*.

$$\begin{aligned}
& \text{ConfirmMempoolTx}(id) \triangleq \\
& \quad \wedge id \in \text{mempool} \\
& \quad \wedge \text{published}[id] = \text{NoSpendHeight} \\
& \quad \wedge \text{LET } tx \triangleq \text{transactions}[id] \\
& \quad \text{IN} \\
& \quad \quad \wedge \text{chain_height}' = \text{chain_height} + 1 \quad \text{Each } tx \text{ is in its own block} \\
& \quad \quad \wedge \text{published}' = [\text{published} \text{ EXCEPT } ![id] = \text{chain_height}'] \\
& \quad \quad \wedge \text{mempool}' = \text{mempool} \setminus \{id\} \\
& \quad \wedge \text{UNCHANGED } \langle \text{transactions} \rangle
\end{aligned}$$

Add a new transaction to *mempool*.

The transaction is created and added to *mempool*.

The transaction is constructed such that it is a valid transaction.

input_type specifies the type of published output to select to spend.

output_type specifies the type of new output to create.

$$\begin{aligned}
& \text{AddSpendTxToMempool}(id, \text{amount}, \text{input_type}, \text{output_type}) \triangleq \\
& \quad \exists \text{spending_output} \in \text{UnspentOutputs} : \\
& \quad \quad \wedge id \notin \text{mempool} \\
& \quad \quad \wedge \text{transactions}[\text{spending_output}[1]].\text{outputs}[\text{spending_output}[2]].\text{type} = \text{input_type} \\
& \quad \quad \wedge \text{transactions}' = [\text{transactions} \text{ EXCEPT } ![id] = \\
& \quad \quad \quad \text{CASE } (\text{output_type} = \text{"p2wkh"}) \rightarrow \\
& \quad \quad \quad \quad \text{CreateP2WKHTx}(\text{spending_output}, \\
& \quad \quad \quad \quad \quad \text{ChooseOutputKeys}(\text{output_type}), \text{amount}) \\
& \quad \quad \quad \square (\text{output_type} = \text{"multisig"}) \rightarrow \\
& \quad \quad \quad \quad \text{CreateMultisigTx}(\text{spending_output}, \\
& \quad \quad \quad \quad \quad \text{ChooseOutputKeys}(\text{output_type}), \text{amount}) \\
& \quad \quad \quad \square (\text{output_type} = \text{"multisig_with_csv"}) \rightarrow \\
& \quad \quad \quad \quad \text{CreateMultisigWithCSVTx}(\text{spending_output}, \\
& \quad \quad \quad \quad \quad \text{ChooseOutputKeys}(\text{output_type}), \text{amount}) \\
& \quad \quad \quad] \\
& \quad \quad \wedge \text{mempool}' = \text{mempool} \cup \{id\} \\
& \quad \wedge \text{UNCHANGED } \langle \text{chain_height}, \text{published} \rangle
\end{aligned}$$