Cardano Cross-chain Transactions

Bridge Addresses

Cardano Testnet

- Using GenerateAsymmetricKeyPair to create a Private/Public key with the Edwards curve for Cardano

In[ ]:=
```
GenerateAsymmetricKeyPair[
 Method → <|"Type" → "EdwardsCurve", "CurveName" → "ed25519"|>]
```

Out[ ]=

```
<|PrivateKey → PrivateKey[ Type: Edwards curve (ed25519) Private key size: 256 b Public key size: 256 b ],
 PublicKey → PublicKey[ Type: Edwards curve (ed25519) Private key size: 256 b Public key size: 256 b ]|>
```

In[ ]:=
```
cardanoBridgeKeys = <|"PrivateKey" → PrivateKey[ Type: Edwards curve (ed25519) Private key size: 256 b Public key size: 256 b ],
   "PublicKey" → PublicKey[ Type: Edwards curve (ed25519) Private key size: 256 b Public key size: 256 b ]|>;
```

- Using BlockchainKeyEncode to encode the previous key to an address

In[ ]:=
```
cardanoBridgeAddress = BlockchainKeyEncode[cardanoBridgeKeys["PublicKey"],
  "Address", BlockchainBase → {"Cardano", "Testnet"}]
```

Out[ ]=

```
addr_test1vp7tuvd0jpq6nc9yxnr2krdnzer0shpsmvrffx5wl06xupq4qzz22
```

- Using an online Testnet faucet (selecting Preprod Testnet)to fund the new address

In[ ]:=
```
SystemOpen["https://developers.cardano.org/en/testnets/cardano/tools/faucet/"]
```

In[ ]:=
```
BlockchainAddressData[cardanoBridgeAddress,
  BlockchainBase → {"Cardano", "Testnet"}] // Dataset
```

Out[ ]=

Ac	addr_test1vp7tuvd0jpq6nc9yxnr2krdnzer0shpsmvrffx5wl06xupq4qzz22
Ba	10 003 000 000 lovelace
As	
To	4
UT	{ ...4 }
To	4
Tr	{ ...4 }

Bitcoin Testnet

- Using GenerateAsymmetricKeyPair to create a Private/Public key with the Elliptic curve for Bitcoin

In[ ]:=
```
GenerateAsymmetricKeyPair["Bitcoin"]
```

Out[ ]=

⟨|PrivateKey → PrivateKey[Type: Elliptic curve (secp256k1) Private key size: 256 b Public key size: 256 b],

PublicKey → PublicKey[Type: Elliptic curve (secp256k1) Public key size: 256 b]|⟩

In[ ]:=
```
btcBridgeKeys = ⟨|"PrivateKey" → PrivateKey[ Type: Elliptic curve (secp256k1) Private key size: 256 b Public key size: 256 b ],
  "PublicKey" → PublicKey[ Type: Elliptic curve (secp256k1) Public key size: 256 b ]|⟩;
```

- Using BlockchainKeyEncode to encode the previous key to an address

In[ ]:=
```
btcBridgeAddress = BlockchainKeyEncode[btcBridgeKeys["PublicKey"],
  "Address", BlockchainBase → {"Bitcoin", "Testnet"}]
```

Out[ ]=

mhq9TEPqsiNNErEEeAZuRGrMvggfYUMBRY

- Using an online Testnet faucet to fund the new address

In[ ]:=
```
SystemOpen["https://coinfaucet.eu/en/btc-testnet/"]
```

In[ ]:=
```
BlockchainAddressData[btcBridgeAddress,
  BlockchainBase → {"Bitcoin", "Testnet"}] // Dataset
```

Out[ ]=

Address	mhq9TEPqsiNNErEEeAZuRGrMvggfYUMBRY
Balance	฿0.0196778
TotalTransactior	3
TransactionList	{ …3 }

Cloud Objects

- Initialize the ADA to BTC Bridge transaction list

In[ ]:=
```
CloudConnect[]
```

Out[ ]=
```
pieros@wolfram.com
```

```
(*CloudPut[{},"Blockchain/tmp/ADA_BTC_bridgeTXs"]*)
```

Out[ ]=
```
CloudObject[
 https://www.wolframcloud.com/obj/pieros/Blockchain/tmp/ADA_BTC_bridgeTXs]
```

- Initialize the BTC to ADA Bridge transaction list

In[ ]:=
```
CloudConnect[]
```

Out[ ]=
```
pieros@wolfram.com
```

```
(*CloudPut[{},"Blockchain/tmp/BTC_ADA_bridgeTXs"]*)
```

In[ ]:=
```
CloudObject[
 https://www.wolframcloud.com/obj/pieros/Blockchain/tmp/BTC_ADA_bridgeTXs]
```

- After reading the Bridge script...That CloudObject is a simple list with transaction hashes that were already executed by the Bridge application

User Addresses

Cardano Testnet

- Using a Cardano Testnet address that already has balance as the user interacting with the bridge

In[ ]:=
```
cardanoUserKeys = <|"PrivateKey" → PrivateKey[ Type: Edwards curve (ed25519) Private key size: 256 b Public key size: 256 b ],
  "PublicKey" → PublicKey[ Type: Edwards curve (ed25519) Private key size: 256 b Public key size: 256 b ]|>;
```

In[ ]:=
```
cardanoUserAddress = BlockchainKeyEncode[cardanoUserKeys["PublicKey"],
  "Address", BlockchainBase → {"Cardano", "Testnet"}]
```

Out[ ]=
```
addr_test1vrg495p8rfxepmv8v8rze6kygqn7ktlvnrvcufsx8x8wj6q004wur
```

Bitcoin Testnet

- Using a Bitcoin Testnet address that already has balance as the user interacting with the bridge

In[ ]:=
```
btcUserKeys = <|"PrivateKey" → PrivateKey[ Type: Elliptic curve (secp256k1) Private key size: 256 b Public key size: 256 b ],
  "PublicKey" → PublicKey[ Type: Elliptic curve (secp256k1) Public key size: 256 b ]|>;
```

In[ ]:=
```
btcUserAddress = BlockchainKeyEncode[btcUserKeys["PublicKey"],
  "Address", BlockchainBase → {"Bitcoin", "Testnet"}]
```

Out[ ]=
```
n2RSh2qDhKq3AMmyzggGYdxfFCVQNdkTch
```

Test Transaction Format (Cardano -> Bitcoin)

- Send a first transaction with the Cross Chain Metadata format

In[ ]:=
```
senderUTXO = First[BlockchainAddressData[cardanoUserAddress,
   "UTXOList", BlockchainBase → {"Cardano", "Testnet"}]]
```

Out[ ]=
```
<|TransactionID → bf382a712fc45adad05ab489970838e730cc10d4c51654f5ac4f29a9dda5c786,
 Index → 1, Amount → 9 989 820 000 lovelace ,
 Tokens → {}, DatumHash → Missing[NotAvailable]|>
```

In[ ]:=
```
fee = Quantity[180 000, "lovelace"];
amount = Quantity[1, "ADA"];
```

In[ ]:=
```
change = senderUTXO["Amount"] - amount - fee
```

Out[ ]=

9.98864×10^{9} lovelace

In[ ]:=
```
txCreate = BlockchainTransaction[<|
   "BlockchainBase" → {"Cardano", "Testnet"},
   "Fee" → Automatic,
   "Inputs" → {
     senderUTXO
    },
   "Outputs" → {
     <|"Address" → cardanoBridgeAddress,
      "Amount" → amount|>,
     <|"Address" → cardanoUserAddress,
      "Amount" → change|>
    },
   "Metadata" → <|15 000 → "Crosschain " <> btcUserAddress|>
   |>]
```

Out[ ]=

BlockchainTransaction[ blockchain base: {Cardano, Testnet } signed: False fee: 180 000 lovelace]

In[ ]:=
```
txSign = BlockchainTransactionSign[txCreate, cardanoUserKeys["PrivateKey"]]
```

Out[ ]=

BlockchainTransaction[ blockchain base: {Cardano, Testnet } signed: True fee: 180 000 lovelace]

In[ ]:=
```
txSubmit = BlockchainTransactionSubmit[txSign]
```
Out[ ]=

BlockchainTransaction[ blockchain base: {Cardano, Testnet } signed: True fee: 180 000 lovelace]

Test Transaction Format (Bitcoin -> Cardano)

- Send a first transaction with the Cross Chain OP_RETURN format

In[ ]:=
```
btcTXList = BlockchainAddressData[btcUserAddress,
  "TransactionList", BlockchainBase → {"Bitcoin", "Testnet"}]
```

In[ ]:=
```
btcUTXO = <|"TransactionID" → First[btcTXList]["TransactionID"],
   "Index" → Position[First[btcTXList]["Outputs"],
       KeyValuePattern["Addresses" → {btcUserAddress}]][[1, 1]] - 1|>;
btcUTXOAmount = First[btcTXList]["Outputs"][[btcUTXO["Index"] + 1]]["Amount"];
```

In[ ]:=
```
btcUTXOAmount
```
Out[ ]=

₿0.0107779

In[ ]:=
```
fee = Quantity[0.00001, "Bitcoins"];
amount = Quantity[0.00002, "Bitcoins"];
```

In[ ]:=
```
change = btcUTXOAmount - amount - fee
```
Out[ ]=

₿0.0107479

In[ ]:=
```
txCreate = BlockchainTransaction[
  <|"BlockchainBase" → {"Bitcoin", "Testnet"},
   "Inputs" → {
     btcUTXO
    },
   "Outputs" → {
     <|"Data" → "Crosschain " <> cardanoUserAddress|>,
     <|"Amount" → amount, "Address" → btcBridgeAddress|>,
     <|"Amount" → change, "Address" → btcUserAddress|>
    }|>]
```
Out[ ]=

BlockchainTransaction[ blockchain base: {Bitcoin, Testnet } signed: False fee: 1000 sat]

In[ ]:=
```
txCreate["Outputs"]
```
Out[ ]=
```
{<|Amount → ฿0 , Data →
   Crosschain addr_test1vrg495p8rfxepmv8v8rze6kygqn7ktlvnrvcufsx8x8wj6q004wur,
  ScriptString → OP_RETURN
     43726f7373636861696e20616464725f74657374317672673439357038726678657 06d76
      387638727a65366b7967716e376b746c766e727663756673783878387 76a36713030347
      77572 |> ,
 <|Amount → ฿0.00002 , Address → mhq9TEPqsiNNErEEeAZuRGrMvggfYUMBRY,
  ScriptString → OP_DUP OP_HASH160
     19610da580ae715dc3ad7baeea023f73308b83a7 OP_EQUALVERIFY OP_CHECKSIG |> ,
 <|Amount → ฿0.0107479 , Address → n2RSh2qDhKq3AMmyzggGYdxfFCVQNdkTch,
  ScriptString → OP_DUP OP_HASH160
     e54fd9810525b2830b4073dd808f20a4cff43e45 OP_EQUALVERIFY OP_CHECKSIG |>}
```

In[ ]:=
```
txSign = BlockchainTransactionSign[txCreate, btcUserKeys["PrivateKey"]]
```
Out[ ]=

BlockchainTransaction[ blockchain base: {Bitcoin, Testnet } signed: True fee: 1000 sat]

In[ ]:=
```
txSubmit = BlockchainTransactionSubmit[txSign]
```
Out[ ]=

BlockchainTransaction[ blockchain base: {Bitcoin, Testnet } signed: True fee: 1000 sat]

Bridge Script

- Add Minimum Fee for easier Cardano Transaction Submit
- Error handling for insufficient balance
- Bridge from BTC to ADA
- UTXO gatherer to increase the amount for a new UTXO

Explanation ADA to BTC

- Getting the latest transaction from addr_test1vp7tuvd0jpq6nc9yxnr2krdnzer0shpsmvrffx5wl06xupq4qzz22 which is the Cardano Bridge address created in the previous section

In[ ]:=
```
cardanoTXID = First[ReverseSortBy[BlockchainAddressData[cardanoBridgeAddress,
    "TransactionList", BlockchainBase → {"Cardano", "Testnet"},
    MaxItems → All], #["Timestamp"] &]]["TransactionID"]
```

Out[ ]=
```
652423938f556965e7c909c1156ff58b46aa1d79ca2f2d40c569f4ca1850ec10
```

In[ ]:=
```
cardanoTX =
 BlockchainTransactionData[cardanoTXID, BlockchainBase → {"Cardano", "Testnet"}]
```

Out[ ]=
```
⟨|TransactionID → 652423938f556965e7c909c1156ff58b46aa1d79ca2f2d40c569f4ca1850ec10,
 BlockHash → 0f4ae0eccd737c9656ccc46225aa5d1a706d26a7573521b208fb77659816855f,
 BlockNumber → 1 254 632, Confirmations → 12, BlockIndex → 1,
 Timestamp → Wed 9 Aug 2023 18:36:17 GMT-5, TotalOutput → 9 989 640 000 lovelace,
 Fee → 180 000 lovelace, Deposit → 0 lovelace, Size → 316 B,
 InvalidHereafter → Missing[NotAvailable], InvalidBefore → Missing[NotAvailable],
 Metadata → ⟨|15000 → Crosschain n2RSh2qDhKq3AMmyzggGYdxfFCVQNdkTch|⟩,
 Mint → {}, Withdrawals → {}, Inputs → {⟨|TransactionID →
     bf382a712fc45adad05ab489970838e730cc10d4c51654f5ac4f29a9dda5c786,
    Index → 1, Amount → 9 989 820 000 lovelace, Address →
     addr_test1vrg495p8rfxepmv8v8rze6kygqn7ktlvnrvcufsx8x8wj6q004wur, Tokens → {},
    Redeemer → Missing[NotAvailable], ScriptHash → Missing[NotAvailable]|⟩},
 Outputs → {⟨|Index → 1, Amount → 9 988 640 000 lovelace,
    Address → addr_test1vrg495p8rfxepmv8v8rze6kygqn7ktlvnrvcufsx8x8wj6q004wur,
    Tokens → {}, Datum → Missing[NotAvailable], DatumHash → Missing[NotAvailable],
    ReferenceScript → Missing[NotAvailable]|⟩,
   ⟨|Index → 0, Amount → 1 000 000 lovelace,
    Address → addr_test1vp7tuvd0jpq6nc9yxnr2krdnzer0shpsmvrffx5wl06xupq4qzz22,
    Tokens → {}, Datum → Missing[NotAvailable], DatumHash → Missing[NotAvailable],
    ReferenceScript → Missing[NotAvailable]|⟩},
 ReferenceInputs → {}, CollateralInputs → {}, CollateralOutputs → {},
 Scripts → {}|⟩
```

- Confirming that the previous transaction has a Metadata format <|“<Crosschain Tag>” -> “Crosschain” + “ <Crosschain address>”|>, in this case the cross chain tag we chose is 15000 and the cross chain address is from Bitcoin.

In[ ]:=
```
KeyExistsQ[cardanoTX["Metadata"], "15000"]
```
Out[ ]=
```
True
```

In[ ]:=
```
{flag, btcDestination} = StringSplit[cardanoTX["Metadata"]["15000"]]
```
Out[ ]=
```
{Crosschain, n2RSh2qDhKq3AMmyzggGYdxfFCVQNdkTch}
```

In[ ]:=
```
SameQ[ToLowerCase@flag, "crosschain"]
```
Out[ ]=
```
True
```

- The following WL code takes the latest transactions received by the Cardano Bridge address
- Then, get the transaction’s Metadata to validate if it’s a cross-chain transaction AND verify if the transaction’s hash is not yet in the list of brigdeTXs
- brigdeTXs is a cloud object that has all the transaction hashes that are applied for the cross-chain operation.
- Then, a new cross-chain transaction is applied with the following properties
- Take the amount in the cross-chain transaction to convert from ADA to BTC
- Build a new transaction with the BTC amount and the BTC Address from the Metadata as an output
- Submit the transaction to the network
- Append the transaction hash to the bridgeTX list

Function ADA to BTC

In[ ]:=
```
bridgeStartADAtoBTC[] := Module[
  {
   cardanoTXID, cardanoTX, flag, cardanoAmount, btcDestination, btcAmount,
   btcTX, btcTXList, btcUTXO, btcUTXOAmount, btcChange, txSubmitted,
   cardanoBridgeAddress =
    "addr_test1vp7tuvd0jpq6nc9yxnr2krdnzer0shpsmvrffx5wl06xupq4qzz22",
```

```
 cardanoBridgeTag = "15000",
 btcBridgeAddress = "mhq9TEPqsiNNErEEeAZuRGrMvggfYUMBRY",

 btcBridgePrivateKey = PrivateKey[ ],
```

Type: Elliptic curve (secp256k1)
Private key size: 256 b
Public key size: 256 b

```
 btcFee = Quantity[0.0001, "Bitcoins"],
 bridgeTXs = CloudGet[CloudObject[
    https://www.wolframcloud.com/obj/pieros/Blockchain/tmp/ADA_BTC_bridgeTXs]]
},

cardanoTXID = First[ReverseSortBy[BlockchainAddressData[cardanoBridgeAddress,
     "TransactionList", BlockchainBase → {"Cardano", "Testnet"},
     MaxItems → All], #["Timestamp"] &]]["TransactionID"];
cardanoTX = BlockchainTransactionData[
  cardanoTXID, BlockchainBase → {"Cardano", "Testnet"}];
{flag, btcDestination} =
 StringSplit[Lookup[cardanoTX["Metadata"], cardanoBridgeTag, ""]];

If[(! MemberQ[bridgeTXs, cardanoTXID]) && SameQ[ToLowerCase@flag, "crosschain"],
 Print["Processing new transaction..."];
 cardanoAmount = SelectFirst[cardanoTX["Outputs"],
    SameQ[#["Address"], cardanoBridgeAddress] &]["Amount"];
 btcAmount = CurrencyConvert[cardanoAmount, "BTC" ];
 btcTXList = BlockchainAddressData[btcBridgeAddress,
   "TransactionList", BlockchainBase → {"Bitcoin", "Testnet"}];
 btcUTXO = <|"TransactionID" → First[btcTXList]["TransactionID"],
   "Index" → Position[First[btcTXList]["Outputs"],
       KeyValuePattern["Addresses" → {btcBridgeAddress}]][[1, 1]] - 1|>;
 btcUTXOAmount = First[btcTXList]["Outputs"][[btcUTXO["Index"] + 1]]["Amount"];
 btcChange = btcUTXOAmount - btcAmount - btcFee;
 btcTX = BlockchainTransaction[
   <|"BlockchainBase" → {"Bitcoin", "Testnet"},
    "Inputs" → {
      btcUTXO
     },
    "Outputs" → {
      <|"Amount" → btcAmount, "Address" → btcDestination|>,
      <|"Amount" → btcChange, "Address" → btcBridgeAddress|>
     }|>];
 Print["Sending transaction..."];
 txSubmitted = BlockchainTransactionSubmit[
```

```
    BlockchainTransactionSign[btcTX, btcBridgePrivateKey]];
  If[! FailureQ[txSubmitted],
   Print[txSubmitted];
   Print[txSubmitted["TransactionID"]];
   CloudPut[Append[bridgeTXs, cardanoTXID],
    "Blockchain/tmp/ADA_BTC_bridgeTXs"];
   Print["Transaction sent. Bridge complete!"]
   ,
   Print["Error submitting transaction..."];
   Print[btcTX]
  ];



  ,
  Print["Waiting for new transactions..."]]
]
```

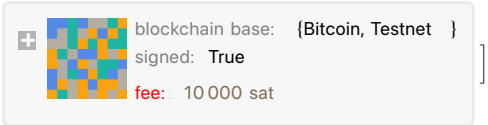
- The following WL code start an schedule task locally to be executed 200 times every 3 seconds

In[ ]:=
```
bridgeStartADAtoBTC[]
```

```
Processing new transaction...

Sending transaction...

BlockchainTransaction[
```




```
]

8d49db0876890578d89fee0be6a393bc90c95412c95b1a38db83fab735b99ef4

Transaction sent. Bridge complete!
```

Explanation BTC to ADA

- Getting the latest transaction from mhq9TEPqsiNNErEEeAZuRGrMvggfYUMBRY which is the Bitcoin Bridge address created in the previous section

In[ ]:=
```
btcTXID = First[BlockchainAddressData[btcBridgeAddress, "TransactionList",
    BlockchainBase → {"Bitcoin", "Testnet"}, MaxItems → All]]["TransactionID"]
```

Out[ ]=
```
0bd27c774eb3138e116901a34990762b29aca7baca81ff6aa6f100ad395f52c5
```

In[ ]:=
```
btcTX = BlockchainTransactionData[btcTXID, BlockchainBase → {"Bitcoin", "Testnet"}]
```

Out[ ]=
```
⟨|TransactionID → 0bd27c774eb3138e116901a34990762b29aca7baca81ff6aa6f100ad395f52c5,
 BlockHash → 0000000000000018d7bf50ecccecf167dc8bc4a7900d98ece5f18c9ce51cacca,
 BlockNumber → 2 474 445, Confirmations → 2, Timestamp → Tue 22 Aug 2023 12:11:16 GMT−5,
 LockTime → 0, Version → 1, TotalInput → ฿0.0107779, TotalOutput → ฿0.0107679,
 Fee → 1000. sat, Size → 310 B, Inputs → {⟨|TransactionID →
     2bf66e443e233d0b63a041e9752056db6b4c988f66bca8ce9342e98210ef9892, Index → 1,
    Amount → ฿0.0107779, ScriptByteArray → ByteArray[106 bytes], ScriptString →
     304402200c315e5bee67092cd8a8b044935d0e5dbae279f0b3c1a83c7ba65dad7ba56f1f022
      06a86a6e21c485b5adbdc8d29b6bab684ecf1b6092b34a2f4b6587eba45e5ad36[ALL
     ]
      023d98022c2f542c9e9cd6c759810c10fda6ffed0d4c728e04aafbd536aad02b9f,
    SequenceNumber → 4 294 967 295, Addresses → {n2RSh2qDhKq3AMmyzggGYdxfFCVQNdkTch},
    SourceConfirmations → 3|⟩},
 Outputs → {⟨|Amount → ฿0., ScriptByteArray → ByteArray[76 bytes],
    ScriptString → OP_RETURN
      43726f7373636861696e20616464725f74657374317672673439357038726678657 0
       6d76387638727a65366b7967716e376b746c766e727663756673783878387 76a367
       1303034777572, Addresses → Missing[NotAvailable],
    SpentQ → False, DestinationTransaction → Missing[NotAvailable]|⟩,
   ⟨|Amount → ฿0.00002, ScriptByteArray → ByteArray[25 bytes],
    ScriptString → OP_DUP OP_HASH160
      19610da580ae715dc3ad7baeea023f73308b83a7 OP_EQUALVERIFY OP_CHECKSIG,
    Addresses → {mhq9TEPqsiNNErEEeAZuRGrMvggfYUMBRY}, SpentQ → False,
    DestinationTransaction → Missing[NotAvailable]|⟩,
   ⟨|Amount → ฿0.0107479, ScriptByteArray → ByteArray[25 bytes],
    ScriptString → OP_DUP OP_HASH160
      e54fd9810525b2830b4073dd808f20a4cff43e45 OP_EQUALVERIFY OP_CHECKSIG,
    Addresses → {n2RSh2qDhKq3AMmyzggGYdxfFCVQNdkTch}, SpentQ → False,
    DestinationTransaction → Missing[NotAvailable]|⟩}|⟩
```

- Confirming that the previous transaction has a Metadata format in the ScriptString of one of the outputs “Crosschain” + “<Crosschain address>” (as a byte hexstring using the OP_RETURN code), in this case the cross chain address is from Cardano.

In[ ]:=
```
metadata = SelectFirst[btcTX["Outputs"],
   StringMatchQ[#["ScriptString"], "OP_RETURN" ~~ __] &]["ScriptString"]
```

Out[ ]=
```
OP_RETURN
  43726f7373636861696e20616464725f746573743176726734393570387266786570 6d763876387
   27a65366b7967716e376b746c766e727663756673783878387 76a3671303034777572
```

In[ ]:=
```
! MissingQ[metadata]
```

Out[ ]=
```
True
```

In[ ]:=
```
{flag, cardanoDestination} = {FromCharacterCode[FromDigits[#, 16] & /@
    StringPartition[StringTake[metadata, {11, 30}], 2]], FromCharacterCode[
   FromDigits[#, 16] & /@ StringPartition[StringDrop[metadata, 32], 2]]}
```

Out[ ]=
```
{Crosschain, addr_test1vrg495p8rfxepmv8v8rze6kygqn7ktlvnrvcufsx8x8wj6q004wur}
```

In[ ]:=
```
SameQ[ToLowerCase@flag, "crosschain"]
```

Out[ ]=
```
True
```

- The following WL code takes the latest transactions received by the Bitcoin Bridge address
- Then, get the transaction's Metadata in one of the outputs to validate if it's a cross-chain transaction AND verify if the transaction's hash is not yet in the list of brigdeTXs
- brigdeTXs is a cloud object that has all the transaction hashes that are applied for the cross-chain operation.
- Then, a new cross-chain transaction is applied with the following properties
- Take the amount in the cross-chain transaction to convert from BTC to ADA
- Build a new transaction with the ADA amount and the Cardano Address from the Metadata as an output
- Submit the transaction to the network
- Append the transaction hash to the bridgeTX list

Function BTC to ADA

In[ ]:=
```
bridgeStartBTCtoADA[] := Module[
  {
```

```
 btcTXID, btcTX, metadata, flag, btcAmount, cardanoDestination,
 cardanoAmount, cardanoTX, cardanoUTXO, cardanoChange, txSubmitted,
 btcBridgeAddress = "mhq9TEPqsiNNErEEeAZuRGrMvggfYUMBRY",
 cardanoBridgeAddress =
  "addr_test1vp7tuvd0jpq6nc9yxnr2krdnzer0shpsmvrffx5wl06xupq4qzz22",

 cardanoBridgePrivateKey = PrivateKey[ 

 ],

 cardanoFee = Quantity[0.18, "ADA"],
 bridgeTXs = CloudGet[CloudObject[
    https://www.wolframcloud.com/obj/pieros/Blockchain/tmp/BTC_ADA_bridgeTXs]]
},

btcTXID = First[BlockchainAddressData[btcBridgeAddress, "TransactionList",
    BlockchainBase → {"Bitcoin", "Testnet"}, MaxItems → All]]["TransactionID"];
btcTX =
 BlockchainTransactionData[btcTXID, BlockchainBase → {"Bitcoin", "Testnet"}];
metadata = SelectFirst[btcTX["Outputs"],
   StringMatchQ[#["ScriptString"], "OP_RETURN" ~~ __] &]["ScriptString"];

If[! MemberQ[bridgeTXs, btcTXID] && ! MissingQ[metadata],
 {flag, cardanoDestination} = {FromCharacterCode[FromDigits[#, 16] & /@
     StringPartition[StringTake[metadata, {11, 30}], 2]], FromCharacterCode[
    FromDigits[#, 16] & /@ StringPartition[StringDrop[metadata, 32], 2]]};

 If[SameQ[ToLowerCase@flag, "crosschain"],
  Print["Processing new transaction..."];
  btcAmount = SelectFirst[btcTX["Outputs"],
     SameQ[#["Addresses"], {btcBridgeAddress}] &]["Amount"];
  cardanoAmount = CurrencyConvert[btcAmount, "ADA" ];
  cardanoUTXO = First[BlockchainAddressData[cardanoBridgeAddress,
     "UTXOList", BlockchainBase → {"Cardano", "Testnet"}]];
  cardanoChange =
   CurrencyConvert[cardanoUTXO["Amount"], "ADA"] - cardanoAmount - cardanoFee;
  cardanoTX = BlockchainTransaction[<|
     "BlockchainBase" → {"Cardano", "Testnet"},
     "Fee" → Automatic,
     "Inputs" → {
       cardanoUTXO
      },
     "Outputs" → {
```

```
          <|"Address" → cardanoDestination,
           "Amount" → cardanoAmount|>,
          <|"Address" → cardanoBridgeAddress,
           "Amount" → cardanoChange|>
         }
        |>];
    Print["Sending transaction..."];
    txSubmitted = BlockchainTransactionSubmit[
      BlockchainTransactionSign[cardanoTX, cardanoBridgePrivateKey]];
    If[! FailureQ[txSubmitted],
     Print[txSubmitted];
     Print[txSubmitted["TransactionID"]];
     CloudPut[Append[bridgeTXs, btcTXID], "Blockchain/tmp/BTC_ADA_bridgeTXs"];
     Print["Transaction sent. Bridge complete!"]
     ,
     Print["Error submitting transaction..."];
     Print[cardanoTX]
    ]
    ,
    Print["Waiting for new transactions..."]
   ]
   ,
   Print["Waiting for new transactions..."]
  ]
 ]
```

The following WL code start an schedule task locally to be executed 200 times every 3 seconds

In[ ]:=
```
bridgeStartBTCtoADA[]
```

```
Processing new transaction...

Sending transaction...

BlockchainTransaction[ blockchain base: {Cardano, Testnet } signed: True fee: 180 001 lovelace ]

9040cd3d9f7f7def027e9d524a756d62d893790a26d9b43c2bfa7ee5ba052867

Transaction sent. Bridge complete!
```

In[ ]:=
```
bridgeStartBTCtoADA[]
```

```
Waiting for new transactions...
```

Example

ADA to BTC

Send ADA to addr_test1vp7tuvd0jpq6nc9yxnr2krdnzer0shpsmvrffx5wl06xupq4qzz22

Metadata: Crosschain n2RSh2qDhKq3AMmyzggGYdxfFCVQNdkTch

- Start the execution of the bridge task once every 20 seconds

In[ ]:=
```
obj = SessionSubmit[ScheduledTask[bridgeStartADAtoBTC[], Quantity[20, "Seconds"]]]
```

Out[ ]=

TaskObject[Task UUID: c98e8b88 –13ec –49da –b6d4 –3213f707ba74
Task environment: Session
Task type: Scheduled
Evaluation expression: bridgeStartADAtoBTC []]

- Send a transaction with the Cross-chain Metadata format

In[ ]:=
```
senderUTXO = First[BlockchainAddressData[cardanoUserAddress,
   "UTXOList", BlockchainBase → {"Cardano", "Testnet"}]]
```

Out[ ]=

```
⟨| TransactionID → 99a7f10697b7b6a598833206b398b7466e07b3b126f027f4530c28ec09e0bcb9,
 Index → 1, Amount → 9 977 280 000 lovelace ,
 Tokens → {}, DatumHash → Missing[NotAvailable] |⟩
```

In[ ]:=
```
fee = Quantity[180 000, "lovelace"];
amount = Quantity[10, "ADA"];
```

In[ ]:=
```
change = senderUTXO["Amount"] - amount - fee
```

Out[ ]=

9.9671×10^{9} lovelace

In[ ]:=
```
txCreate = BlockchainTransaction[<|
   "BlockchainBase" → {"Cardano", "Testnet"},
   "Fee" → Automatic,
   "Inputs" → {
     senderUTXO
    },
   "Outputs" → {
     <|"Address" → cardanoBridgeAddress,
      "Amount" → amount|>,
     <|"Address" → cardanoUserAddress,
      "Amount" → change|>
    },
   "Metadata" → <|15 000 → "Crosschain " <> btcUserAddress|>
   |>]
```

Out[ ]=

BlockchainTransaction[ blockchain base: {Cardano, Testnet } signed: False fee: 180 000 lovelace]

In[ ]:=
```
txSign = BlockchainTransactionSign[txCreate, cardanoUserKeys["PrivateKey"]]
```

Out[ ]=

BlockchainTransaction[ blockchain base: {Cardano, Testnet } signed: True fee: 180 000 lovelace]

In[ ]:=
```
txSubmit = BlockchainTransactionSubmit[txSign]
```

Out[ ]=

BlockchainTransaction[ blockchain base: {Cardano, Testnet } signed: True fee: 180 000 lovelace]

■ Suspend the execution of the bridge task if needed

In[ ]:=
```
TaskSuspend[obj]
```

Out[ ]=

TaskObject[Task UUID: c98e8b88 −13ec −49da −b6d4 −3213f707ba74 Task environment: Session Task type: Scheduled Evaluation expression: bridgeStartADAtoBTC []]

BTC to ADA

Send BTC to mhq9TEPqsiNNErEEeAZuRGrMvggfYUMBRY

Metadata: Crosschain addr_test1vrg495p8rfxepmv8v8rze6kygqn7ktlvnrvcufsx8x8wj6q004wur

Start the execution of the bridge task once every 30 seconds

In[ ]:=
```
obj = SessionSubmit[ScheduledTask[bridgeStartBTCtoADA[], Quantity[30, "Seconds"]]]
```

Out[ ]=

TaskObject[Task UUID: 576d741b –ac5a –4f01 –9ae5 –3b8b9211f81c
Task environment: Session
Task type: Scheduled
Evaluation expression: bridgeStartBTCtoADA []
]

Send a transaction with the Cross Chain Metadata structure

In[ ]:=
```
btcTXList = BlockchainAddressData[btcUserAddress,
   "TransactionList", BlockchainBase → {"Bitcoin", "Testnet"}];
```

In[ ]:=
```
btcUTXO = <|"TransactionID" → First[btcTXList]["TransactionID"],
   "Index" → Position[First[btcTXList]["Outputs"],
       KeyValuePattern["Addresses" → {btcUserAddress}]][[1, 1]] - 1|>;
btcUTXOAmount = First[btcTXList]["Outputs"][[btcUTXO["Index"] + 1]]["Amount"];
```

In[ ]:=
```
btcUTXO
```

Out[ ]=

<|TransactionID → 0bd27c774eb3138e116901a34990762b29aca7baca81ff6aa6f100ad395f52c5, Index → 2|>

In[ ]:=
```
btcUTXOAmount
```

Out[ ]=

฿0.0107479

In[ ]:=
```
fee = Quantity[0.00001, "Bitcoins"];
amount = Quantity[0.00002, "Bitcoins"];
```

In[ ]:=
```
change = btcUTXOAmount - amount - fee
```

Out[ ]=

฿0.0107179

In[ ]:=
```
txCreate = BlockchainTransaction[
  <|"BlockchainBase" → {"Bitcoin", "Testnet"},
   "Inputs" → {
     btcUTXO
    },
   "Outputs" → {
     <|"Data" → "Crosschain " <> cardanoUserAddress|>,
     <|"Amount" → amount, "Address" → btcBridgeAddress|>,
     <|"Amount" → change, "Address" → btcUserAddress|>
    }|>]
```

Out[ ]=

BlockchainTransaction[ blockchain base: {Bitcoin, Testnet } signed: False fee: 1000 sat]

In[ ]:=
```
txSign = BlockchainTransactionSign[txCreate, btcUserKeys["PrivateKey"]]
```

Out[ ]=

BlockchainTransaction[ blockchain base: {Bitcoin, Testnet } signed: True fee: 1000 sat]

In[ ]:=
```
txSubmit = BlockchainTransactionSubmit[txSign]
```

Out[ ]=

BlockchainTransaction[ blockchain base: {Bitcoin, Testnet } signed: True fee: 1000 sat]

Suspend the execution of the bridge task if needed

In[ ]:=
```
TaskSuspend[obj]
```

Out[ ]=

TaskObject[Task UUID: 576d741b −ac5a −4f01 −9ae5 −3b8b9211f81c
Task environment: Session
Task type: Scheduled
Evaluation expression: bridgeStartBTCtoADA []]