Address Lookup Tables

Address Lookup Tables, commonly referred to as "lookup tables " or "ALTs " for short, allow developers to create a collection of related addresses to efficiently load more addresses in a single transaction.

Since each transaction on the Solana blockchain requires a listing of every address that is interacted with as part of the transaction, this listing would effectively be capped at 32 addresses per transaction. With the help ofAddress Lookup Tables , a transaction would now be able to raise that limit to 256 addresses per transaction.

Compressing on chain addresses#

After all the desired addresses have been stored on chain in an Address Lookup Table, each address can be referenced inside a transaction by its 1-byte index within the table (instead of their full 32-byte address). This lookup method effectively "compresses " a 32-byte address into a 1-byte index value.

This "compression " enables storing up to 256 addresses in a single lookup table for use inside any given transaction.

Versioned Transactions#

To utilize an Address Lookup Table inside a transaction, developers must use v0 transactions that were introduced with the newVersioned Transaction format .

How to create an address lookup table#

Creating a new lookup table with the@solana/web3.js library is similar to the olderlegacy transactions, but with some differences.

Using the@solana/web3.js library, you can use thecreateLookupTable function to construct the instruction needed to create a new lookup table, as well as determine its address:

const web3 = require("@solana/web3.js");

// connect to a cluster and get the currentslot const connection = new web3.Connection(web3.clusterApiUrl("devnet")); const slot = await connection.getSlot();

// Assumption: // payer is a valid Keypair with enough SOL to pay for the execution

const [lookupTableInst, lookupTableAddress] = web3.AddressLookupTableProgram.createLookupTable({ authority: payer.publicKey, payer: payer.publicKey, recentSlot: slot, });

console.log("lookup table address:", lookupTableAddress.toBase58());

// To create the Address Lookup Table on chain: // send thelookupTableInst instruction in a transaction NOTE: Address lookup tables can becreated with either av0 transaction or alegacy transaction. But the Solana runtime can only retrieve and handle the additional addresses within a lookup table while usingv0 Versioned Transactions .

Add addresses to a lookup table#

Adding addresses to a lookup table is known as "extending ". Using the@solana/web3.js library, you can create a newextend instruction using theextendLookupTable method:

// add addresses to the lookupTableAddress table via an extend instruction const extendInstruction = web3.AddressLookupTableProgram.extendLookupTable({ payer: payer.publicKey, authority: payer.publicKey, lookupTable: lookupTableAddress, addresses: [payer.publicKey, web3.SystemProgram.programId, // list more publicKey addresses here], });

// Send this extendInstruction in a transaction to the cluster // to insert the listing ofaddresses into your lookup table with address lookupTableAddress NOTE: Due to the same memory limits oflegacy transactions, any transaction used toextend an Address Lookup Table is also limited in how many addresses can be added at a time. Because of this, you will need to use multiple transactions toextend any table with more addresses (~20) that can fit within a single transaction's memory limits. Once these addresses have been inserted into the table, and stored on chain, you will be able to utilize the Address Lookup Table in future transactions. Enabling up to 256 addresses in those future transactions.

Fetch an Address Lookup Table#

Similar to requesting another account (or PDA) from the cluster, you can fetch a complete Address Lookup Table with

thegetAddressLookupTable method:

// define the PublicKey of the lookup table to fetch const lookupTableAddress = new web3.PublicKey("");

// get the table from the cluster const lookupTableAccount = (await connection.getAddressLookupTable(lookupTableAddress)).value;

// lookupTableAccount will now be a AddressLookupTableAccount object

console.log("Table address from cluster:", lookupTableAccount.key.toBase58()); OurlookupTableAccount variable will now be aAddressLookupTableAccount object which we can parse to read the listing of all the addresses stored on chain in the lookup table:

// loop through and parse all the addresses stored in the table for (let i = 0; i < lookupTableAccount.state.addresses.length; i++) { const address = lookupTableAccount.state.addresses[i]; console.log(i, address.toBase58()); }

How to use an address lookup table in a transaction#

After you have created your lookup table, and stored your needed address on chain (via extending the lookup table), you can create av0 transaction to utilize the on chain lookup capabilities.

Just like olderlegacy transactions, you can create all theinstructions your transaction will execute on chain. You can then provide an array of these instructions to theMessage used in the `v0 transaction.

NOTE: The instructions used inside av0 transaction can be constructed using the same methods and functions used to create the instructions in the past. There is no required change to the instructions used involving an Address Lookup Table. // Assumptions: // - arrayOfInstructions has been created as an array of TransactionInstruction // - we are using thelookupTableAccount obtained above

// construct a v0 compatible transaction Message const messageV0 = new web3.TransactionMessage({ payerKey: payer.publicKey, recentBlockhash: blockhash, instructions: arrayOfInstructions, // note this is an array of instructions }).compileToV0Message([lookupTableAccount]);

// create a v0 transaction from the v0 message const transactionV0 = new web3.VersionedTransaction(messageV0);

// sign the v0 transaction using the file system wallet we created named payer transactionV0.sign([payer]);

// send and confirm the transaction // (NOTE: There is NOT an array of Signers here; see the note below...) const txid = await web3.sendAndConfirmTransaction(connection, transactionV0);

console.log(Transaction: https://explorer.solana.com/tx{txid}?cluster=devnet,); NOTE: When sending aVersionedTransaction to the cluster, it must be signed BEFORE calling thesendAndConfirmTransaction method. If you pass an array ofSigner (like withlegacy transactions) the method will trigger an error!

More Resources#

- Read theproposal
- for
- Address Lookup Tables and Versioned transactions
- Example Rust program using Address Lookup Tables