Solana RPC Methods & Documentation

Interact with Solana nodes directly with the JSON RPC API via the HTTP and Websocket methods.

Configuring State Commitment#

For preflight checks and transaction processing, Solana nodes choose which bank state to query based on a commitment requirement set by the client. The commitment describes how finalized a block is at that point in time. When querying the ledger state, it's recommended to use lower levels of commitment to report progress and higher levels to ensure the state will not be rolled back.

In descending order of commitment (most finalized to least finalized), clients may specify:

- finalized
-
    - the node will query the most recent block confirmed by
- supermajority of the cluster as having reached maximum lockout, meaning the
- cluster has recognized this block as finalized
- confirmed
-
    - the node will query the most recent block that has been voted on
- by supermajority of the cluster.* It incorporates votes from gossip and replay.
-
    - It does not count votes on descendants of a block, only direct votes on that
-
    - block.
-
    - This confirmation level also upholds "optimistic confirmation" guarantees in
-
    - release 1.3 and onwards.
- processed
-
    - the node will query its most recent block. Note that the block
- may still be skipped by the cluster.

For processing many dependent transactions in series, it's recommended to useconfirmed commitment, which balances speed with rollback safety. For total safety, it's recommended to usefinalized commitment.

Default Commitment#

If commitment configuration is not provided, the node will default tofinalized commitment

Only methods that query bank state accept the commitment parameter. They are indicated in the API Reference below.

RpcResponse Structure#

Many methods that take a commitment parameter return an RpcResponse JSON object comprised of two parts:

- context
- : An RpcResponseContext JSON structure including aslot
- field at
- which the operation was evaluated.
- value
- : The value returned by the operation itself.

Parsed Responses#

Some methods support anencoding parameter, and can return account or instruction data in parsed JSON format if"encoding":"jsonParsed" is requested and the node has a parser for the owning program. Solana nodes currently support JSON parsing for the following native and SPL programs:

Program Account State Instructions Address Lookup v1.15.0 v1.15.0 BPF Loader n/a stable BPF Upgradeable Loader stable stable Config stable SPL Associated Token Account n/a stable SPL Memo n/a stable SPL Token stable stable SPL Token 2022 stable stable Stake stable stable Vote stable stable The list of account parsers can be foundhere , and instruction parsershere .

Filter criteria#

Some methods support providing afilters object to enable pre-filtering the data returned within the RpcResponse JSON object. The following filters exist:

- memcmp: object
-
  - compares a provided series of bytes with program account
- data at a particular offset. Fields:
-
  - offset: usize
-
  -
    - offset into program account data to start comparison
-
  - bytes: string
-
  -
    - data to match, as encoded string
-
  - encoding: string
-
  -
    - encoding for filterbytes
-
  - data, either "base58" or
-
  - "base64". Data is limited in size to 128 or fewer decoded bytes.
-
  - NEW: This field, and base64 support generally, is only available in
-
  - solana-core v1.14.0 or newer. Please omit when querying nodes on earlier
-
  - versions
- dataSize: u64
-
  - compares the program account data length with the provided
- data size