Checkpoint

Checkpoints contain finalized transactions and are used for node synchronization and global transaction ordering.

```
type Checkpoint {
  digest: String!
  sequenceNumber: UInt53!
  timestamp: DateTime!
  validatorSignatures: Base64!
  previousCheckpointDigest: String
  networkTotalTransactions: UInt53
  rollingGasSummary: GasCostSummary
  epoch: Epoch
  transactionBlocks(
    first: Int
    after: String
    last: Int
    before: String
    filter: TransactionBlockFilter
    scanLimit: Int
  ): TransactionBlockConnection!
  bcs: Base64
}
```

Fields

`Checkpoint.digest.String!` non-null scalar

A 32-byte hash that uniquely identifies the checkpoint contents, encoded in Base58. This hash can be used to verify checkpoint contents by checking signatures against the committee, Hashing contents to match digest, and checking that the previous checkpoint digest matches.

`Checkpoint.sequenceNumber.UInt53!` non-null scalar

This checkpoint's position in the total order of finalized checkpoints, agreed upon by consensus.

`Checkpoint.timestamp.DateTime!` non-null scalar

The timestamp at which the checkpoint is agreed to have happened according to consensus. Transactions that access time in this checkpoint will observe this timestamp.

Checkpoint.validatorSignatures.Base64! non-null scalar

This is an aggregation of signatures from a quorum of validators for the checkpoint proposal.

Checkpoint.previousCheckpointDigest.String scalar

The digest of the checkpoint at the previous sequence number.

Checkpoint.networkTotalTransactions.UInt53 scalar

The total number of transaction blocks in the network by the end of this checkpoint.

Checkpoint.rollingGasSummary.GasCostSummary object

The computation cost, storage cost, storage rebate, and non-refundable storage fee accumulated during this epoch, up to and including this checkpoint. These values increase monotonically across checkpoints in the same epoch, and reset on epoch boundaries.

Checkpoint.epoch.Epoch object

The epoch this checkpoint is part of.

Checkpoint.transactionBlocks.TransactionBlockConnection! non-null object

Transactions in this checkpoint.

`scanLimit` restricts the number of candidate transactions scanned when gathering a page of results. It is required for queries that apply more than two complex filters (on function, kind, sender, recipient, input object, changed object, or ids), and can be at most `serviceConfig.maxScanLimit`.

When the scan limit is reached the page will be returned even if it has fewer than `first` results when paginating forward (`last` when paginating backwards). If there are more transactions to scan, `pageInfo.hasNextPage` (or `pageInfo.hasPreviousPage`) will be set to `true`, and `PageInfo.endCursor` (or `PageInfo.startCursor`) will be set to the last transaction that was scanned as opposed to the last (or first) transaction in the page.

Requesting the next (or previous) page after this cursor will resume the search, scanning the next `scanLimit` many transactions in the direction of pagination, and so on until all transactions in the scanning range have been visited.

By default, the scanning range consists of all transactions in this checkpoint.

Checkpoint.transactionBlocks.first.Int scalar

Checkpoint.transactionBlocks.after.String scalar

Checkpoint.transactionBlocks.last.Int scalar

Checkpoint.transactionBlocks.before.String scalar

Checkpoint.transactionBlocks.filter.TransactionBlockFilter input

Checkpoint.transactionBlocks.scanLimit.Int scalar

Checkpoint.bcs.Base64 scalar

The Base64 serialized BCS bytes of CheckpointSummary for this checkpoint.

Returned By

checkpoint query

Member Of

AvailableRange object . CheckpointConnection object . CheckpointEdge object
. TransactionBlockEffects object

Edit this page