Mempool Data Program

This program makes Blocknative's Mempool Data Archive available for the benefit of the community.

About the Data

Blocknative actively maintains the most comprehensive historical dataset of mempool transaction events within the Ethereum ecosystem. This collection contains >8TB of archive data representing >16 billion transaction detection events since November 1st, 2019.

It is updated daily at 13 UTC time with a typical update containing 11M events for a 0.012TB size, though the heaviest days on the network can be as large as 41M events and 0.3TB size.

This uninterrupted dataset covers major scenarios the network has encountered over the years, including massive surges in traffic, huge gas spikes, bidding wars, the launch of MEV-boost, the price of ETH collapsing, EIP-1559, Black Thursday, and major hacks.

This data covers 27 data fields, such as gas details, input data, time pending in the mempool, failure reasons, and regional timestamps for each instance seen by our global network of nodes.

Our self-operated infrastructure provides the earliest detection times from North America, Asia, and Europe.

The Mempool Archive set is available for research and may be freely used for non-commercial purposes.

To download the archive, visitthis page on ethernow.xyz.

Data Schema

Blocknative logs all mempool transactions from nodes in multiple geographical regions for the Ethereum mainnet blockchain. The Archive contains historic events for all transactions:

- entering the mempool
- denied entry into the mempool (rejection with reason)
- exiting the mempool (eviction with reason)
- replacing existing mempool transaction (speedup or cancel)
- finalized on chain (confirmed or failed)
-

The number of times a transaction appears in the Archive corresponds to the number of status changes it undergoes. The detecttime

field indicates the time when the status change was first observed.

Below you can find the complete schema for the data:

Field Name Description Data Type Example detecttime Timestamp that the transaction was detected in mempool. TIMESTAMP 2020-03-12 00:00:00.409000 hash Unique identifier hash for a given transaction. VARCHAR (66) 0x6b4104838fd153b2d1ab705737843f5ea99666794391dd52653960970dc7e5ef status Status of the transaction. VARCHAR (66) Pending ,speedup ,cancel ,failed ,stuck ,dropped ,confirmed ,evicted ,rejected region The geographic region for the node that detected the transaction. VARCHAR (66) us-east-1 ,eu-central-1 ,ap-southeast-1 reorg If there was a reorg, refers to the blockhash of the reorg. VARCHAR (66) 0xf2ec4b2a7b951e4400e99d1171c4fb875fd388b15b6cb97bf5ad1c8dbea3a73a replace If the transaction was replaced (speedup/cancel), the transaction hash of the replacement. VARCHAR (66) 0xcea6244a7f0a7c2630085ca3e47e1ecfc28a5c03a08a8f3ec5f43fbef3d83dd5 curblocknumber The block number the event was detected in. NUMERIC (18) 12429202 failurereason If a transaction failed, this field provides contextual information. VARCHAR (66) Reverted: ""UniswapV2Router: INSUFFICIENT_OUTPUT_AMOUNT"" blockspending If a transaction was finalized (confirmed, failed), this refers to the number of blocks that the transaction was waiting to get on-chain. INT 2 timepending If a transaction was finalized (confirmed, failed), this refers to the time in milliseconds that the transaction was waiting to get on-chain. BIGINT 4678 nonce A unique number which counts the number of transactions sent from a given address. NUMERIC (38) 27744 gas The maximum number of gas units allowed for the transaction. NUMERIC (38) 55588 gasprice The price offered to the miner/validator per unit of gas. Denominated in wei. NUMERIC (38) 1200000000 value The amount of ETH transferred or sent to contract. Denominated in wei. NUMERIC (38) 147940000000000 toaddress The destination of a given transaction. VARCHAR (42) 0x501c885e8f519feeb1a8f9429ea586ebd378b549 fromaddress The source/initiator of a given transaction. VARCHAR (42) 0xf974334a62b3aab3e2b5509f65b9b2141d8efa03 input Additional data that can be attached to a transaction. This field can be used to tell a smart contract to execute a function. VARCHAR (65535) 0xa9059cbb000000000000000000000000955a0ef4e120528f8486c04c97388d530cfbf23900000000000000000000 network The specific Ethereum network used. VARCHAR (32) bsc-main ,goerli ,kovan ,main ,rinkeby ,ropsten ,xdai type Post EIP-1559, this indicates how the gas parameters are submitted to the network: - type 0 - legacy

- type 1 - usage of access lists according to EIP-2930

- type 2 - usingmaxpriorityfeepergas andmaxfeepergas INT 0 ,1 ,2 maxpriorityfeepergas The maximum value for a tip offered to the miner/validator per unit of gas. The actual tip paid can be lower if (maxfee -basefee) <maxpriorityfee . Denominated in wei. NUMERIC (38) 111373960022 maxfeepergas The maximum value for the transaction fee (includingbasefee and tip) offered to the miner/validator per unit of gas. Denominated in wei. NUMERIC (38) 111373960022 basefeepergas The fee per unit of gas paid and burned for thecurblocknumber . This fee is algorithmically determined. Denominated in wei. NUMERIC (38) 111373960022 dropreason If the transaction was dropped from the mempool, this describes the contextual reason for the drop. VARCHAR (26) unexecutable-txs ,unpayable-txs ,replaced-txs ,account-cap-txs ,old-txs ,underpriced-txs ,low-nonce rejectionreason If the transaction was rejected from the mempool, this describes the contextual reason for the rejection. VARCHAR (64) exceeds block gas limit ,insufficient funds for gas * price + value intrinsic gas too low ,non transaction ,underpriced stuck A transaction was detected in the queued area of the mempool and is not eligible for inclusion in a block. BOOLEAN 1 gasused If the transaction was published on-chain, this value indicates the amount of gas that was actually consumed. Denominated in wei. INT 111373960022 detect_date A truncated version ofdetecttime . Best used as a partition for large datasets and as a search parameter to speed up queries. VARCHAR (64) 2023-10-10

Data Access

Free, public mempool data is limited to academic and non-commercial purposes with attribution to Blocknative. For commercial use please fill outthis form to contact us directly.

Use Cases

The Mempool Data Program can be used to research:

- Historic gas trends
- Censorship
- Transaction inclusion
- MEV
- Private transactions
- Bug fixes
- Block sequences of interest
- Trading strategies
- Third-party strategies
- Malicious activity
- Probes for potential explotation
-

Frequently Asked Questions

What attribution must I provide when using the Blocknative Mempool Data?

The archive is publicly available according to open data standards and licenses datasets under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International.

2.1 Attribution — End Users must give appropriate credit, provide a link to the license, and indicate if changes were made. End Users may do so in any reasonable manner, but not in any way that suggests the licensor endorses End Users or their use.2.2 NonCommercial — End Users may not use the material for commercial purposes.2.3 ShareAlike — If End Users remix, transform, or build upon the material, End Users must distribute their contributions under the same license as the original.

Please use the following as a guideline for attribution:

1. Papers
2. : Data provided byBlocknative
3. ?
4. ?
5.

If you have any questions please reach out to us orDiscord .

What format is the data?

The data is stored in hourly slices with file format*.csv.gz The data is tab delimited.

How many nodes are gathering mempool data?

We run highly redundant node infrastructure in each region to ensure strong uptime.

How can I identify on-chain transactions?

On-chain transaction have aconfirmed status.

```
Copy SELECT* FROMmempool_archive WHEREstatus='confirmed'
```

How can I identify private transactions?

A private transaction does not have apending event.timepending is determined from the difference between a transaction'spending event andconfirmed event.

```
Copy SELECT* FROMmempool_archive WHEREtimepending=0 ANDstatus='confirmed'
```

What is the difference between dropreason and rejectionreason ?

A dropped transaction might have been valid but deemed less important or lower-priority. A rejected transaction is one that is fundamentally flawed or invalid according to the Ethereum protocol rules.

A drop reason could be that there isn't enough ETH in the EOA to cover gas fees. A rejection reason could be incorrect transaction signatures.

Dropped transactions existed in the mempool, but are dropped to make room for incoming transactions. Rejected transactions never make it to the mempool.

Previous Notify (older version) Next 4337 Tools Last updated2 months ago

Was this helpful?