D4D4D4;--ch-t-background: #1E1E1E;--ch-t-lighter-inlineBackground: #1e1e1ee6;--ch-t-editor-background: #1E1E1E;--ch-t-editor-foreground: #D4D4D4;--ch-t-editor-rangeHighlightBackground: #ffffff0b;--ch-t-editor-infoForeground: #3794FF;--ch-t-editor-selectionBackground: #264F78;--ch-t-focusBorder: #007FD4;--ch-t-tab-activeBackground: #1E1E1E;--ch-t-tab-activeForeground: #ffffff;--ch-t-tab-inactiveBackground: #2D2D2D;--ch-t-tab-inactiveForeground: #ffffff80;--ch-t-tab-border: #252526;--ch-t-tab-activeBorder: #1E1E1E;--ch-t-editorGroup-border: #444444;--ch-t-editorGroupHeader-tabsBackground: #252526;--ch-t-editorLineNumber-foreground: #858585;--ch-t-input-background: #3C3C3C;--ch-t-input-foreground: #D4D4D4;--ch-t-icon-foreground: #C5C5C5;--ch-t-sideBar-background: #252526;--ch-t-sideBar-foreground: #D4D4D4;--ch-t-sideBar-border: #252526;--ch-t-list-activeSelectionBackground: #094771;--ch-t-list-activeSelectionForeground: #fffffe;--ch-t-list-hoverBackground: #2A2D2E; }

Running the Safe Transaction Service

The Safe Transaction Service tracks transactions sent via Safe contracts. It indexes these transactions using events (L2 chains) and tracing (L1 chains) mechanisms.

Key Features

- Blockchain indexing
- : Executed transactions, configuration changes, ERC-20/721 transfers, and on-chain confirmations are automatically indexed from the blockchain.
- Off-chain transaction signatures
- : Transactions can be sent to the service, enabling off-chain signature collection. This feature helps inform owners about pending transactions awaiting confirmation for execution.
- Off-chain messages
- : The service can collect off-chain signatures to confirm messages followingEIP-1271(opens in a new tab)
- .
- Transaction decoding
- : The service keeps getting source and ABIs from contracts that interact with Safe to decode these interactions.

Technology stack overview

The Safe Transaction Service is aDjango(opens in a new tab) app written in Python with the following architecture:

- Gunicorn(opens in a new tab)
- : A Python WSGI HTTP Server.
- Celery(opens in a new tab)
- : A task queue with focus on real-time processing, while also supporting task scheduling. The Safe Transaction Service has a scheduler (for periodic tasks), a worker indexer to consume and execute indexing tasks, and a contracts worker mainly to get metadata from contracts.
- RabbitMQ(opens in a new tab)
- : A distributed message broker system Celery uses to share messages between the scheduler, workers, and the Django

application.
- PostgreSQL(opens in a new tab)
- : An open source object-relational database system.
- Redis(opens in a new tab)
- : An open source, in-memory data structure store used for caching by the Safe Transaction Service.
- safe-eth-py(opens in a new tab)
- : A library to interact with Safe and blockchains.

Blockchain indexing

The Safe Transaction Service can index automatically executed transactions, configuration changes, ERC-20/721 transfers, and on-chain confirmations. The indexer is running onworker-indexer by different periodictasks(opens in a new tab) .

ERC-20 and ERC-721 are indexed usingeth_getLogs (opens in a new tab) filtered by the Transfer topickeccak('Transfer(address,address,uint256)') .

Safe creation, executed transactions, configuration changes, and on-chain confirmations are indexed differently depending on whether the chain is L1 or L2.

For L1 chains, the indexer calls tracing methods. For the oldest blockstrace_filter (opens in a new tab) is used filtering by singleton address of Safe contracts, and for the latest blockstrace_block (opens in a new tab) is used, astrace_filter takes longer to return updated information.trace_block will be used if the block depth is lower thanETH_INTERNAL_TXS_NUMBER_TRACE_BLOCKS . The environment variables indexing uses are definedhere(opens in a new tab) .

For L2 chains, the indexing is by events with theeth_getLogs (opens in a new tab) method with the corresponding topics.

From Safe creation, the Transaction Service stores each contract change on theSafeStatus model asnonce ,owners , etc. The latest and current status of a Safe is stored asSafeLastStatus for easy database access and optimization.

Endpoints

The following endpoints show the current indexing status of the Safe Transaction Service:

- GET /v1/about/indexing/

Response example:

_10 { _10 "currentBlockNumber": 9773327, // Last block on the blockchain _10 "erc20BlockNumber": 9773326, // Last block indexed for erc20/721 events _10 "erc20Synced": true, _10 "masterCopiesBlockNumber": 9773327, // Last block indexed for executed transactions, ether transfers, configuration changes, etc. _10 "masterCopiesSynced": true, _10 "synced": true _10 }

Reorgs handling

Every block is marked asnot confirmed during indexing unless it has some depth (configured via theETH_REORG_BLOCKS environment variable). Unconfirmed blocks are checked periodically to see if the blockchainblockHash for that number changed before it reaches the desired number of confirmations. If that's the case, all blocks from that block and related transactions are deleted, and indexing is restarted to the last confirmed block.

Note: No off-chain signatures, transactions, or messages are lost in this process. Only on-chain data is removed.

Off-chain transaction signatures

The Safe Transaction Service can collect off-chain transaction signatures, allowing the owners to share their signatures to reach the required threshold before executing a transaction and spending less gas than on-chain approvals.

Endpoints

The following endpoints let us propose a transaction and collect every confirmation (off-chain signatures):

- POST /v1/safes/{address}/multisig-transactions/
- : Creates a new transaction and requires at least one signature.
- POST /v1/multisig-transactions/{safe_tx_hash}/confirmations/
- : Adds a new confirmation and needs thesafe_tx_hash
- .
- GET /v1/multisig-transactions/{safe_tx_hash}/
- : Returns all the multi-signature transaction information.
- GET /v1/multisig-transactions/{safe_tx_hash}/confirmations/
- : Returns the list of all confirmations to a multi-signature transaction.

The following sequence diagram shows a use case for a Safe shared by Alice and Bob where at least one confirmation for each one is required:

What's thesafe_tx_hash ?

safe_tx_hash is the unique identifier for a Safe transaction and is calculated using theEIP-712(opens in a new tab) standard: keccak256(0x19 || 0x1 || domainSeparator || safeTxHashStruct)

wheresafeTxHashStruct is thehashStruct of a Safe transaction.

The followingGuides show how to create and execute Safe transactions from the Safe Transaction Service.

Off-chain messages

The Safe Transaction Service can collect the necessary off-chain signatures to confirm a message using EIP-1271(opens in a new tab) . The message can be a string (EIP-191 is used to get the hash) or an object EIP-712.

Endpoints

- GET /v1/safes/{address}/messages/
- : Returns the messages created for the given Safe address.
- POST /v1/safes/{address}/messages/
- : Creates a message with at least one signature.
- GET /v1/messages/{message_hash}/
- : Returns a message for a given message hash.
- POST /v1/messages/{message_hash}/signatures/
- : Adds another signature to the message with the given message hash.

The following sequence diagram shows a use case for a Safe shared by Alice and Bob where at least one signature for each one is required to confirm a message fully:

The followingGuides show how to create and sign messages from the Safe Transaction Service.

Transaction decoder

The Safe Transaction Service can decode contract interactions. To achieve it, the service periodically gets source and ABIs from different sources like Sourcify, Etherscan, and Blockscout using thesafe-eth-py library.

The detection of contract interactions is done in a periodic task executed every hour formultisig-transaction andmodule-transactions or every six hours formultisend-transactions onworker-contracts-tokens . For every new contract, the service tries to download the source, and the ABI requests it first to Sourcify, then Etherscan, and as a last chance, Blockscout. It's important to know that not all these data sources are supported or configured for every network onsafe-eth-py .

Supported and configured networks onsafe-eth-py :

- Sourcify supported networks(opens in a new tab)
- Etherscan configured networks(opens in a new tab)
- Blockscout configured networks(opens in a new tab)

Endpoints

- POST /v1/data-decoder/
- decodes a transactiondata
- passed on the body for ato
- contract address.

Example transaction decoder

_10 curl -X 'POST' \ _10 'https://safe-transaction-sepolia.safe.global/api/v1/data-decoder/' \ _10 -H 'accept: application/json' \ _10 -H 'Content-Type: application/json' \ _10 -H 'X-CSRFToken: Gx1aRa8kIJGIAfReLAWwr9Q6dHv22dFt7VprdipLryHcxpfhk9aV0UDAhNz8gGYz' \ _10 -d '{ _10 "data": "0x095ea7b3000000000000000000000000e6fc577e87f7c977c4393300417dcc592d90acf8ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff", _10 "to": "0x4127839cdf4F73d9fC9a2C2861d8d1799e9DF40C" _10 }'

Output:

_15 { _15 "method": "approve", _15 "parameters": [_15 { _15 "name": "spender", _15 "type": "address", _15 "value": "0xe6fC577E87F7c977c4393300417dCC592D90acF8" _15 }, _15 { _15 "name": "value", _15 "type": "uint256", _15 "value": "1157920892373161954235709850086879078532699846656405640394575840079131296399 35" _15 } _15] _15 }

This decoded data is also included asdataDecoded in theGET /multisig-transactions ,GET /module-transactions andGET /all-transactions endpoints.

Running and Maintenance

In theSafe Transaction Service(opens in a new tab) GitHub repository, under theSetup for production (event indexing)(opens in a new tab) andSetup for production (tracing mode)(opens in a new tab)sections, it is shown how to run the service and retrieve

events from any supported network type. Note that the configuration must be adapted to the needs of a specific use case.

In addition, the Service maintenance(opens in a new tab) section shows how to deal with some common errors in the service operation.

Overview RPC Requirements