Understanding Bundles

Searchers use Flashbots to submit bundles to block builders for inclusion in blocks. Bundles are one or more transactions that are grouped together and executed in the order they are provided. In addition to the searcher's transaction(s) a bundle can also potentially contain other users' pending transactions from the mempool, and bundles can target specific blocks for inclusion as well. Here's an example:

const blockNumber =

await provider . getBlockNumber ( ) const minTimestamp =

(await provider . getBlock (blockNumber)) . timestamp const maxTimestamp = minTimestamp +

120 const signedBundle = flashbotsProvider . signBundle ([{ signedTransaction :

SIGNED_ORACLE_UPDATE_FROM_PENDING_POOL

// serialized signed transaction hex } , { signer : wallet ,

// ethers signer transaction : transaction // ethers populated transaction object }]) const bundleReceipt =

await flashbotsProvider . sendRawBundle (signedBundle ,

// bundle we signed above targetBlockNumber ,

// block number at which this bundle is valid { minTimestamp ,

// optional minimum timestamp at which this bundle is valid (inclusive) maxTimestamp ,

// optional maximum timestamp at which this bundle is valid (inclusive) revertingTxHashes :

[tx1 , tx2]

// optional list of transaction hashes allowed to revert. Without specifying here, any revert invalidates the entire bundle. }) In the above example we've constructed a bundle that includes our transaction (transaction) and a transaction from the mempool: SIGNED_ORACLE_UPDATE_FROM_PENDING_POOL. Edit this page Last updatedonJan 30, 2024 Previous Rust Provider Next Send transaction/bundle to multiple builders