Sending Bundles

While the newbundle spec may look complex, a bundle can be sent with default parameters using a client library, which ends up looking quite similar to sending bundles on mev-boost.

Sending a Bundle

Placeholders Variables inALL_CAPS are placeholders. To use this code, replace them with your own data. * mev-share-client-ts

import MevShareClient ,

{ BundleParams }

from

'@flashbots/mev-share-client'

const mevShareClient = MevShareClient . useEthereumMainnet (authSigner)

// ...

const bundle =

[{ hash :

PENDING_TX_HASH } , { tx :

await wallet . signTransaction (BACKRUN_TX) , canRevert :

false } ,]

const params : BundleParams =

{ inclusion :

{ block : targetBlock , // target several blocks withmaxBlock maxBlock : targetBlock +

3 , } , body : bundle , }

const bundleResult =

await mevShareClient . sendBundle (params) The first transaction in the bundle contains the hash of a pending transaction, which we presumably got from listening to theevent stream . The second transaction is one we sign ourself, which backruns the first transaction, presumably for a profit.

Bundle endpoint MEV-Share bundles are sent to the same RPC endpoint as MEV-Boost bundles:https://relay.flashbots.net Seemev-share-client-ts for a full working example.

The params not included in the previous example can be used to configure more advanced conditions for your bundle's execution and the MEV refunds you earn.

Share bundle data

To earn more MEV refunds and improve your bundle's chances of inclusion, you can specify data about your bundle's transactions that you wish to share with other searchers. The data that is shared is specified via theprivacy parameter.

bundles with signed txs only Note that bundles with transactions including thehash parameter are considered "unmatched", and will throw an error. * mev-share-client-ts * curl

const params : BundleParams =

{ inclusion :

{ block :

17539448 , maxBlock :

17539450 } , body :

[{ tx :

"0x02..." , canRevert :

false } , { tx :

"0x02..." , canRevert :

false } ,] , privacy :

{ hints :

{ calldata :

true , logs :

true , txHash :

true , } , } , } curl -X POST -H "Content-Type: application/json" -H "X-Flashbots-Signature: FB_SIG" -d '{ inclusion: { block: "0x10ba178", maxBlock: "0x10ba17a", }, body: [{tx: "0x02...", canRevert: false}, {tx: "0x02...", canRevert: false},], privacy: { hints: ["calldata", "logs", "tx_hash"], }, }' Even sharing only thetx_hash hint can be enough to earn some extra MEV. The only requirement to earn it is that your bundle is included in another searcher's bundle.

This strategy is particularly relevant to searchers who operate on public mempool transactions, and/or employ strategies that leave leftover MEV on the table (for example, if your strategy changes prices enough for a backrun arbitrage to be profitable).

Nest bundles (be a MEV-Share Node)

MEV-Share Nodes nest bundles to build composite bundles that are more profitable.

- mev-share-client-ts

const params : BundleParams =

{ inclusion :

{ block :

TARGET_BLOCK , } , body :

{ { hash :

PENDING_TX_HASH } , { bundle :

{ inclusion :

{ ... } , body :

{ ... } , privacy :

{ ... } , validity :

{ ... } , } } } , }

Builder Inheritance

By default, when a user specifies which builders their transactions should be sent to (by settingbuilders), those settings are inherited by bundles which use those transactions.

Searchers can restrict these settings by specifyingbuilders in the bundle'sprivacy parameters. Specifically, by settingbuilders , the bundle is sent to theintersection of all builders specified by each of the bundle's transactions, and the builders specified in the bundle'sbuilders parameter.

Simulating Bundles

Bundles matched by the MEV-Share Node can be simulated to check MEV profits. To simulate a bundle, we use themev_simBundle endpoint.

only matched bundles can be simulated Bundle simulations can only be executed on matched bundles, which contain only signed transactions, or nested bundles which also contain only signed transactions. Bundles with transactions including thehash parameter are considered "unmatched", and will throw an error. * mev-share-client-ts

const params : BundleParams =

{ inclusion :

{ block :

TARGET_BLOCK } , body :

[{ bundle :

{ version :

"beta-1" , inclusion :

{ block :

TARGET_BLOCK , maxBlock :

TARGET_BLOCK

+

3 , } , body :

[{ tx :

SIGNED_TX_1 }] , privacy :

{ hints :

{ calldata :

true } , } , validity :

{ refundConfig :

[{ address :

REFUND_ADDRESS , percent :

100 }] } } } , { tx :

SIGNED_TX_2 } ,] , validity :

{ refund :

[{ bodyIdx :

0 , percent :

90 }] } }

const simResult =

await mevShareClient . simulateBundle (params)Edit this page Last updatedonFeb 13, 2024 Previous Understanding Bundles Next Debugging