Understanding Bundles

Bundles on MEV-Share are conceptually the same as bundles on MEV-Boost: they are an ordered array of transactions that execute atomically. However, their structure is a bit different. MEV-Share bundles use a new method calledmev_sendBundle which has additional fields used to specify privacy preferences and introduce other new features like post-execution validity checks.

Bundle Definition

MEV-Share Bundles have the following structure:

/ *NOTE: optional fields are marked with a ? example: { privacy?: { ... } // privacy is optional }*/ { jsonrpc :

"2.0" , id :

string

|

number , method :

"mev_sendBundle" , params :

[{

/ *MevSendBundleParams* / version :

"v0.1" , inclusion :

{ block :

string ,

// hex-encoded number maxBlock ? :

string ,

// hex-encoded number } , body :

Array < { hash :

string

}

| { tx :

string , canRevert :

boolean

}

| { bundle : MevSendBundleParams }

, validity ? :

{ refund ? :

Array < { bodyIdx :

number , percent :

number , }

, refundConfig ? :

Array < { address :

string , percent :

number , }

} , privacy ? :

{ hints ? :

Array < "calldata"

| "contract_address"

| "logs"

| "function_selector"

| "hash"

| "tx_hash"

, builders ? :

Array < string

, } , metadata ? :

{ originId ? :

string , } }] } Key Fields:

- inclusion
- : Defines the pre-inclusion predicates to check (e.g. block range).
- body
- : Contains transactions, bundle hashes, or transaction hashes.
- validity
- : (Optional) Defines the post-inclusion predicates to check, which are just refund parameters at the moment.
- privacy
- : (Optional) Sets privacy configurations, including which builders to submit to. NOTE: the Flashbots builder is submitted to by default even if only other builders are specified.
- Optional properties are denoted with a ?.

Note to searchers on builders 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. This is the generic bundle structure used in MEV-Share. This comprehensive specification enables several exciting features, outlined in the next section.

Sharing hints

MEV-Share bundles can share hints with searchers, which can be used to backrun the bundle. This is done by setting theprivacy parameter inmev_sendBundle . Theprivacy parameter is an object with the following fields:

Hint Description calldata Share data sent to the smart contract (if applicable) by the transaction. The function selector and contract address will also be shared if the calldata is shared. logs Share logs emitted by executing the transaction. default_logs Share specific subset of logs related to defi swaps. Partial info (the pool id and the fact that a swap was made) for curve, balancer, and uniswapV2/V3-style trades function_selector Share the 4-byte identifier of the function being called on the smart contract by the transaction. The contract address will also be shared if the function selector is shared. contract_address Share the address of the recipient of the transaction; typically a smart contract. hash Share the transaction hash (or bundle hash if sending a bundle). To use full privacy mode, share this hint and this hint alone. The hash will always be shared if other hints are shared. tx_hash Share individual tx hashes in the bundle. Searchers can share hints to give other searchers information about their bundle that would allow them to be backrun. If your bundle gets backrun by another searcher, you get paid a cut of the MEV they extract!

Builders

MEV-Share bundles can be sent to multiple builders at once. This is done by setting thebuilders field in theprivacy parameter ofmev_sendBundle .

Bundle composition (backrunning other bundles)

With theprivacy parameter inmev_sendBundle you can share select information about your bundle with other searchers, who can then use that to try to extract MEV. Should they succeed, you get paid some of the MEV they extracted!

One example that works well with bundle composition is a liquidation bot. Liquidations often cause a price shift, leaving MEV on the table which can be captured by arbitrage with a backrun bundle. If you run a liquidation bot, you can earn more MEV by sending your bundles to MEV-Share with thetx_hash hint enabled, which will allow other searchers to backrun your bundle.

An example would look something like this:

- mev-share-client-ts

const params : BundleParams =

{ inclusion :

{ block :

17539448 , maxBlock :

17539458 } , body :

[{ tx :

"0x02..." , canRevert :

false } , { tx :

"0x02..." , canRevert :

false } ,] , privacy :

{ hints :

{ txHash :

true , } , } , } Specifying thetx_hash hint in your bundle shares the hashes of your bundle's transactions with searchers on MEV-Share, which is what allows them to backrun your bundle. Again, when they do this, you earn a cut of the profit!

You may also try experimenting with other hints to give searchers more data with which to formulate a backrun. Sharing more data will lower your privacy, but will make your bundle easier to backrun, and increase the likelihood of your bundles earning extra MEV.

only original transactions are supported Bundles that set theprivacy parameter can only contain original signed transactions in thebody parameter. Bundles using transactions specified by{hash} are not allowed to use the privacy parameter (full bundle privacy is maintained in this case). Allowing such bundles to share data using theprivacy parameter would compromise the privacy guarantees of user transactions.

SeeSending Bundles

for more information.

Now that we know all the different ways in which we can send and share bundles, we're finally ready tosend a bundle . Edit this page Last updatedonJan 30, 2024 Previous Event Stream Next Sending Bundles