Package Mempool Accept

Definitions

A **package** is an ordered list of transactions, representable by a connected Directed Acyclic Graph (a directed edge exists between a transaction that spends the output of another transaction).

For every transaction `t` in a **topologically sorted** package, if any of its parents are present in the package, they appear somewhere in the list before `t`.

A **child-with-unconfirmed-parents** package is a topologically sorted package that consists of exactly one child and all of its unconfirmed parents (no other transactions may be present). The last transaction in the package is the child, and its package can be canonically defined based on the current state: each of its inputs must be available in the UTXO set as of the current chain tip or some preceding transaction in the package.

Package Mempool Acceptance Rules

The following rules are enforced for all packages:

- Packages cannot exceed `MAX_PACKAGE_COUNT=25` count and `MAX_PACKAGE_SIZE=101KvB` total size (#20833)
    - *Rationale*: This is already enforced as mempool ancestor/descendant limits. If transactions in a package are all related, exceeding this limit would mean that the package can either be split up or it wouldn't pass individual mempool policy.
    - Note that, if these mempool limits change, package limits should be reconsidered. Users may also configure their mempool limits differently.
- Packages must be topologically sorted. (#20833)
- Packages cannot have conflicting transactions, i.e. no two transactions in a package can spend the same inputs. Packages cannot have duplicate transactions. (#20833)
- No transaction in a package can conflict with a mempool transaction. BIP125 Replace By Fee is currently disabled for packages. (#20833)
    - Package RBF may be enabled in the future.
- When packages are evaluated against ancestor/descendant limits, the union of all transactions' descendants and ancestors is considered. (#21800)
    - *Rationale*: This is essentially a "worst case" heuristic intended for packages that are heavily connected, i.e. some transaction in the package is the ancestor or descendant of all the other transactions.

The following rules are only enforced for packages to be submitted to the mempool (not enforced for test accepts):

- Packages must be child-with-unconfirmed-parents packages. This also means packages must contain at least 2 transactions. (#22674)
    - *Rationale*: This allows for fee-bumping by CPFP. Allowing multiple parents makes it possible to fee-bump a batch of transactions. Restricting packages to a defined topology is easier to reason about and simplifies the validation logic greatly.

    - Warning: Batched fee-bumping may be unsafe for some use cases. Users and application developers should take caution if utilizing multi-parent packages.

- Transactions in the package that have the same txid as another transaction already in the mempool will be removed from the package prior to submission ("deduplication").
    - *Rationale*: Node operators are free to set their mempool policies however they please, nodes may receive transactions in different orders, and malicious counterparties may try to take advantage of policy differences to pin or delay propagation of transactions. As such, it's possible for some package transaction(s) to already be in the mempool, and there is no need to repeat validation for those transactions or double-count them in fees.
    - *Rationale*: We want to prevent potential censorship vectors. We should not reject entire packages because we already have one of the transactions. Also, if an attacker first broadcasts a competing package or transaction with a mutated witness, even though the two same-txid-different-witness transactions are conflicting and cannot replace each other, the honest package should still be considered for acceptance.

Package Fees and Feerate

Package Feerate is the total modified fees (base fees + any fee delta from `prioritisetransaction`) divided by the total virtual size of all transactions in the package. If any transactions in the package are already in the mempool, they are not submitted again ("deduplicated") and are thus excluded from this calculation.

To meet the two feerate requirements of a mempool, i.e., the pre-configured minimum relay feerate (`minRelayTxFee`) and the dynamic mempool minimum feerate, the total package feerate is used instead of the individual feerate. The individual transactions are allowed to be below the feerate requirements if the package meets the feerate requirements. For example, the parent(s) in the package can pay no fees but be paid for by the child.

Rationale: This can be thought of as "CPFP within a package," solving the issue of a parent not meeting minimum fees on its own. This would allow contracting applications to adjust their fees at broadcast time instead of overshooting or risking becoming stuck or pinned.

Rationale: It would be incorrect to use the fees of transactions that are already in the mempool, as we do not want a transaction's fees to be double-counted.

Implementation Note: Transactions within a package are always validated individually first, and package validation is used for the transactions that failed. Since package feerate is only calculated using transactions that are not in the mempool, this implementation detail affects the outcome of package validation.

Rationale: Packages are intended for incentive-compatible fee-bumping: transaction B is a "legitimate" fee-bump for transaction A only if B is a descendant of A and has a *higher* feerate than A. We want to prevent "parents pay for children" behavior; fees of parents should not help their children, since the parents can be mined without the child. More generally, if transaction A is not needed in order for transaction B to be mined, A's fees cannot help B. In a child-with-parents package, simply excluding any parent transactions that meet feerate requirements individually is sufficient to ensure this.

Rationale: We must not allow a low-feerate child to prevent its parent from being accepted; fees of children should not negatively impact their parents, since they are not necessary for the parents to be mined. More generally, if transaction B is not needed in order for transaction A to be mined, B's fees cannot harm A. In a child-with-parents package, simply validating parents individually first is sufficient to ensure this.

Rationale: As a principle, we want to avoid accidentally restricting policy in order to be backward-compatible for users and applications that rely on p2p transaction relay. Concretely, package validation should not prevent the acceptance of a transaction that would otherwise be policy-valid on its own. By always accepting a transaction that passes individual validation before trying package validation, we prevent any unintentional restriction of policy.