Unconditional inclusion lists

[

upload_d8336517ff27bb713b162aea8e862611

2648×2220 258 KB

](https://ethresear.ch/uploads/default/original/2X/d/d6fe238dfb55e13146a0b91b64a514c48122f44d.jpeg)

^lol

\cdot

by mike & toni based on discussions with potuz & terence

\cdot

tuesday – january 30, 2024

\cdot

tl;dr;

Inclusion lists return agency to the proposer in a PBS world by allowing them to express preferences over the transactions included onchain, despite outsourcing their block production to the specialized builder market. This document discretizes the inclusion list design, advocates for a specific feature set, and aims to justify these decisions based on an analysis of the tradeoffs.

\cdot

Acks

Many thanks to Roman, Dan, Barnabé, Francesco, Julian, Thomas, Stokes, Hsiao-Wei, Justin, Vitalik, & Mikhail for thoughtful comments and reviews!

\cdot

Related work

Article

Description

[No free lunch

](https://ethresear.ch/t/no-free-lunch-a-new-inclusion-list-design/16389)

research post

[IL eip

](https://eips.ethereum.org/EIPS/eip-7547)

eip

[censorship.pics

](https://censorship.pics/)

toni’s dashboard

[Spec’ing out ILs

](https://ethresear.ch/t/specing-out-forward-inclusion-list-w-dedicated-gas-limits/17115)

terence’s research post

Non-expiring inclusion lists

toni’s research post

draft spec pr

hsiao-wei’s draft CL spec change

High-level summary

We advocate for unconditional inclusion lists.

The two-slot process is:

1. The slot n

proposer builds an inclusion list published alongside their block.

1. The transactions in the inclusion list are either:

a) included in the slot n

execution payload,

b) included in the slot n+1

execution payload, or

c) appended (in order) to the end of the slot n+1

execution payload.

1. The validity of the slot n+1

block depends on the existence and satisfaction of a signed inclusion list summary.

The following figure demonstrates this process.

[

upload_9e2c321a76fd4bf927abb937e65284e4|522x500,

1463×1401 275 KB

](https://ethresear.ch/uploads/default/original/2X/d/d13e0e4550ac6d8037eff026875129ba99c78dbc.png)

Note that this is slightly different than the original proposal in [“No free lunch”

](https://ethresear.ch/t/no-free-lunch-a-new-inclusion-list-design/16389), in which transaction inclusion is conditioned on sufficient gas remaining in the slot n+1

block. Instead, the unconditional IL acts as a “block extension” appended to the slot n+1

payload. The figure below shows the transaction order that results from this design:

[

upload_b0e3599e2c43335cb956bdde3a4b75af-1

461×727 37.3 KB

](https://ethresear.ch/uploads/default/original/2X/c/cbd1c5da2dfed2f654010529b134661caff2fdac.png)

While unconditional ILs require a selection for the “inclusion list gas limit”, this value is not the focus of this document. The remainder of this document aims to justify the other elements of this design.

Core design philosophy.

The “forward” property

– This property dictates the enforcement of the slot n

inclusion list during slot n+1

. This has been a core desideratum in inclusion list designs since Francescopublished the idea at the end of 2022. The forward property provides incentive compatibility because the slot n

proposer can safely construct an IL with the knowledge that the value of their block will be unimpacted by the IL.

The “unconditional” property

– This property ensures that once an IL includes a transaction, that transaction is guaranteed to go onchain at the latest in the subsequent slot (unless another transaction with the same nonce from the same address takes its place, see here for the details).

The “bottom-of-next-block” property

– This property enforces any slot n

IL transactions not in the main body of either the slot n

or slot n+1

payloads must be included at the end of the slot n+1

payload. Note that these transactions could also be included ~at the top~ of the slot n+1

block. The top-of-block vs bottom-of-block remains one of the main sticking points in design discussion; we highlight the tradeoffs below.

The “ordered” property

– This property mandates any slot n

IL transactions not in the main body of either the slot n

or slot n+1

payloads must be included in the order at the bottom of the slot n+1

payload. In other words, any IL transactions occupying the “bottom-of-next-block” must appear in the same order as the IL.

By separating these four criteria we independently address the natural questions that arise from each. We skip the “forward” property because it is discussed elsewhere ad nauseam.

[unconditional property] How should inclusion list enforcement be handled?

- The enforcement of the IL depends on how you view the IL object in the context of the main block body. The two ideas below capture the difference:
- The slot n

IL extends the slot n+1

block. (unconditional)

The IL has a fixed gas limit and the execution of the transactions occurs if they are not in the slot n

payload. These transactions consume the IL gas (as opposed to the slot n+1

30mm gas limit). (Note that viewing unconditional IL transactions as an extension of the gas limit of a block could equivalently be seen as a reduction of the gas limit of a block if the transactions are force-included even if the block is full.)

1. The slot n

IL constrains the slot n+1

block. (conditional)

In this case, the IL does not have any extra gas beyond the 30mm contained in the payload, thus conditioning the enforcement of the IL transactions on the existence of extra unused gas in the slot n+1

payload.

- The slot n

IL extends the slot n+1

block. (unconditional)

The IL has a fixed gas limit and the execution of the transactions occurs if they are not in the slot n

payload. These transactions consume the IL gas (as opposed to the slot n+1

30mm gas limit). (Note that viewing unconditional IL transactions as an extension of the gas limit of a block could equivalently be seen as a reduction of the gas limit of a block if the transactions are force-included even if the block is full.)

- The slot n

IL constrains the slot n+1

block. (conditional)

In this case, the IL does not have any extra gas beyond the 30mm contained in the payload, thus conditioning the enforcement of the IL transactions on the existence of extra unused gas in the slot n+1

payload.

[unconditional property] What are the tradeoffs between unconditional and conditional ILs?

- Pros of unconditional ILs
- The IL transactions are guaranteed to be included in the next block, no matter what the payload contains.

Because the IL serves as an extension of the gas limit of the block, we have maximally strong assurances about transaction inclusion.

- The IL transactions are guaranteed to be included in the next block, no matter what the payload contains.

Because the IL serves as an extension of the gas limit of the block, we have maximally strong assurances about transaction inclusion.

- Cons of unconditional ILs
- The IL becomes a de-facto block size increase, which could induce an external market for IL construction.

During periods of high demand for block space, getting included in the IL may be extremely valuable, making MEV over the IL relevant.

- The IL becomes a de-facto block size increase, which could induce an external market for IL construction.

During periods of high demand for block space, getting included in the IL may be extremely valuable, making MEV over the IL relevant.

- Pros of conditional ILs
- The IL provides no strict guarantees and is less MEV-able.

This is the inverse of the previous point. When the IL transactions are only conditioned on the block having remaining gas, there is much less value gained from getting included in the IL during high congestion periods. However, because of EIP-1559, blocks are not regularly full (only 11690 / 432970 \approx 2.7\%

of blocks contain >29mm gas over the last 3 months), so this may not significantly change the demand for IL inclusion.

- The IL provides no strict guarantees and is less MEV-able.

This is the inverse of the previous point. When the IL transactions are only conditioned on the block having remaining gas, there is much less value gained from getting included in the IL during high congestion periods. However, because of EIP-1559, blocks are not regularly full (only 11690 / 432970 \approx 2.7\%

of blocks contain >29mm gas over the last 3 months), so this may not significantly change the demand for IL inclusion.

- Cons of conditional ILs
- The transaction inclusion guarantees are weaker because the ILs are conditional.

As a result, a censoring builder might choose to fill the block with irrational transactions to avoid using the IL (see Barnabé's post [Fun and Games

](https://ethresear.ch/t/fun-and-games-with-inclusion-lists/16557) for the description of "block stuffing"). With conditional guarantees, multiple proposers may have to include the transaction in the IL before the transaction is included in a block (e.g. if the first block is full).

- The transaction inclusion guarantees are weaker because the ILs are conditional.

As a result, a censoring builder might choose to fill the block with irrational transactions to avoid using the IL (see Barnabé's post [Fun and Games

](https://ethresear.ch/t/fun-and-games-with-inclusion-lists/16557) for the description of "block stuffing"). With conditional guarantees, multiple proposers may have to include the transaction in the IL before the transaction is included in a block (e.g. if the first block is full).

[bottom-of-next-block property] Why include the IL transactions at the bottom of the slot n+1

payload (as a suffix) instead of at the top (as a prefix)?

- The top-of-block positioning is valuable. Consider the case where an NFT mint starts at slot n+1

. MEV searchers who want to be the first to mint the NFT can guarantee this privilege by bribing the slot n

proposer to construct an IL with their transactions in it. This may induce a marketplace (similar to mev-boost

) for proposers to auction off the right to construct an IL for their slot. Externalization of the IL is not desirable because we want the validators to construct their ILs based on their local view of the mempool to preserve the censorship resistance benefits of having a decentralized validator set. With the IL transactions at the end of the slot n+1

payload, these transactions have no guarantees beyond inclusion. The downside is that IL transactions may be arbitrarily front-ran by transactions in the slot n+1

payload. However, this doesn't significantly change the semantics of getting in an IL because even in the slot n+1

top-of-block version, IL transactions can still be indiscriminately frontran by transactions in the slot n

payload.

[bottom-of-next-block property] Is there a situation where the guarantee of being included in the end-of-block for slot n+1

is valuable?

- We don't think so; consider the reframing of the question: "Is there value in having an arbitrarily 'frontrunable' end-of-block transaction?" Both the slot n

and slot n+1

builders can arbitrarily insert transactions at the end of their block to exploit the IL transactions. In that regard, these transactions don't have any guarantees about the post-state of slot n+1

. Further, they must be broadcast by t=4

in slot n

, meaning they have no latency advantage over just being included in the slot n

or slot n+1

payloads directly.

[ordered property] Why do the transactions in the IL get inserted in order?

- Great question, simple answer: there does not seem to be any reason to allow their reordering. Preserving the IL ordering simplifies the implementation (de-duplicate the IL \rightarrow

linearly scan) and reduces the control the slot n+1

builder can exert over the slot n

IL.

Inclusion list usage considerations.

How do we know that proposers will use inclusion lists?

- We don't know. As with the rest of the validator specification, we rely on honest participants to run one of the default client implementations. These defaults will construct the IL before proposing their blocks. Some users may choose to modify their client software to behave differently. On a higher level, we are asking: "Who do we want to depend on for censorship resistance of the protocol?" The status quo depends on builders who have total autonomy over what transactions end up in the block; this has obvious downsides as the builders are a small set of doxed actors. ILs allow validators to reclaim some agency over their block space and bring the censorship resistance properties back to the validator set, which we assume to be more decentralized and much longer-tailed than the builders. One amazing feature of inclusion lists is that they don't rely on an "honest majority" assumption; even a small minority (e.g., 5% of validators) using ILs greatly hardens the protocol censorship resistance (though we would hope for a higher adoption rate of course).

How do ILs interact with self-building?

- ILs do not change the process of self-building a block. When the self-building proposer receives the previous block over the network, an IL accompanies it. The self-building proposer inserts the transactions in the IL as the suffix to their block (they could also just as easily include the transactions in the main block body). Most commonly, the self-building proposer will have an empty inclusion list for the subsequent slot because they will include all visible transactions in their block. If the self-built block is full, they can fill their IL with the next highest-paying transactions (greedy).

How could we incentivize inclusion list usage?

- To make the IL usage more meaningful, we may consider simple options to incentivize validators to opt in. For example, if we encourage validators to put all the transactions they see into an IL (in effect constructing a locally built block), we could reward them through the tips of all the included transactions (even if the transaction inclusions are in the subsequent slot payload). While we think this would increase IL usage generally, validators could still choose to exclude a small subset of transactions they want to censor without sacrificing much in terms of priority fees. Thus it is not clear that this would meaningfully improve the censorship resistance of the protocol. We advocate for starting with unincentivized ILs and only consider incentivizing them in the future if initial adoption is insufficient and more data is available.

Are there legal concerns with IL usage?

- This is not legal advice; I am not a lawyer.

Some node operators may modify the client software to exclude a subset of transactions. Despite this ILs can ~only improve~ the censorship resistance of the protocol. Consider that right now, validators self-select into three groups. 1. Validators who connect to non-censoring relays/builders.

1. Validators who *do not* connect to non-censoring relays/builders.
2. Validators who self-build.
3. Validators who connect to non-censoring relays/builders.
4. Validators who *do not* connect to non-censoring relays/builders.
5. Validators who self-build.
6. With ILs, we now have four groups (assuming here that validators in group 2 above will not use ILs).

\;\;

1a. Validators who connect to non-censoring relays/builders and construct ILs.

\;\;

1b. Validators who connect to non-censoring relays/builders and*do not* construct ILs.

\;\;

1. Validators who *do not* connect to non-censoring relays/builders and *do not* construct ILs.

\;\;

1. Validators who self-build.
2. Note that we get the improvement of censorship resistance from group (1a) above because even though those validators may end up proposing a censored block (e.g. if a censoring builder has the highest paying block), they produce an IL that will be enforced on the subsequent slot. Group (1b) continues to connect to non-censoring relays and builders, so they don't increase the amount of censoring that they do, they just don't contribute to the censorship resistance of the protocol because they don't construct ILs.

Additional design considerations.

How large should the inclusion list be?

- If we go for an unconditional inclusion list that acts as an extension to the gas limit of a block, we would need to choose the size of that limit accordingly. In the most congested periods, this will set an upper bound on the total gas limit of the block + IL pair. For example, an IL with 10mm gas would mean the total block could be 40mm, effectively raising the gas limit. We defer further discussion on the exact gas limit of the IL to future work.

Should the inclusion lists be expiring or non-expiring?

- An additional design axis to consider is if the inclusion list has an “expiration”. Expiration enforces that the IL enforcement occurs within some specified window of slots. For example, should an IL expire after just one slot, even the missed slot scenario, or should it be enforced on the next proposer regardless? A non-expiring design comes with the advantage of not allowing validators with subsequent proposers to deliberately miss the first slot if they prefer not to adhere to the IL to enforce their censorship. However, missing the slot intentionally to avoid IL conformance is a dramatic decision. In the current design by Potuz, the IL doesn’t expire and remains valid even after a missed slot. With a stricter approach to enforcement, we eliminate the possibility of intentionally missing the first of several consecutive slots to circumvent the IL. For a more detailed discussion on this topic, see the thread under this post.

Should the inclusion lists be cumulative?

- This design consideration only matters if we want proposers to be able to specify a certain number of slots in the future by which then the IL ~must~ be enforced. For example, a transaction that pays enough fees to be included in any of the next m

blocks, even if the base fee rises continuously until then, could be put onto the IL with the deadline of m

. This implies that the subsequent proposer is not obligated to adhere to the IL unless the deadline for inclusion is the current slot. A notable downside of this approach is that transaction inclusion may have to wait for a few more slots. The advantage of such a design is that it prevents certain games that entities with consecutive slots could play. For example, an entity with a 10% validator share has three consecutive slots ~7 times per day. This entity could leave the IL empty if the subsequent proposer is its own, but fill the IL if the subsequent proposer belongs to someone else. This strategy allows the entity to avoid constraining its proposers while continuing to impose constraints on others.