Contract Compatibility

How to make your contract compatible with Relay

Relay allows a user to pay on Chain A for an arbitrary set of transactions to be executed on Chain B. These transactions are executed by a relayer, not the user themselves, which means you need to make special considerations to make sure your contract is compatible.

General overview

When Relayers execute txs on behalf of users, they do so via a multicall contract (to be specific, we use Vectorized's gas-optimizedMulticaller contract). This protects the Relayer from malicious transactions.

One exception is simple bridge transactions, where thedata is empty. For these, the Relayer will execute them directly, in order to avoid gas overhead of going via Multicaller.

Sender Attribution

Because transactions are executed via the Relayer / Multicaller, you can't rely onmsg.sender for authentication. There are a couple of ways around this:

Unauthenticated Delegation

The simplest solution is if your contract allows one user to take an action on behalf of another user. This works great for actions that don't require authentication, because they are purely net beneficial to the user:

- buying an NFT
- depositing into a vault
- placing an auction bid

You just need to allow passing the address that the action is being done on behalf of. Of course, you can't do this for actions that potentially harm the user:

- selling the NFT
- withdrawing from the vault
- canceling the bid, etc.

Some contracts don't require authentication, but also don't have native delegation built into them. In such scenarios, it might be possible to use a router contract. E.g. if an NFT only supported minting tomsg.sender , you could mint via a router contract then have it forwarded the NFT to the user.

Authenticated Delegation

For authenticated actions, you can do those by passing in proof that the user authenticated the action. Currently, you need to build custom signature authentication into your contract, however, in the near future we will be adding support for more standardized flows, such as:

- Selling tokens with permit signatures
- MulticallerWithSigner
- ERC2771 Trusted Forwarder

Just-In-Time Gas

Alternatively, you can simply bridge a small amount of gas and have the user execute the transaction themselves. This is viable because of how fast and cheap Relay is, and is backwards compatible with any contract.

We will soon add functionality to our SDK to handle this flow completely automatically, but until then, you can do it manually like this:

- Estimate the cost of the tx on destination
- Get a quote to bridge enough gas (small buffer recommended to handle fluctuation)
- User executes the bridge of gas money from Origin to Destination
- User executes the action on Destination
- (Optional) If selling an asset, User could potentially bridge the proceeds back to Origin

This might seem like a lot, but especially when Origin and Destination are both L2s, it can be quite fast and cheap. Arguably, much better than the user needing to worry about having a balance on many chains. Cross-Chain Execution UX Best Practices twitter Powered by Mintlify

- General overview
- Sender Attribution
- Unauthenticated Delegation
- Authenticated Delegation
- Just-In-Time Gas