

Why 7683 is broken and how to fix it

Not a diss track

Vaibhav Chellani (aka VC)

Founder, Socket Protocol

ERC-7683: Cross Chain Intents Standard

■ ERCs erc-20, standards-adoption, cross-chain, interop



marktoda

2 Apr 11

The Cross Chain Intents Standard is meant to unify off-chain messages and on-chain settlement smart contracts to enable sharing of infrastructure, filler networks, and orders across cross-chain bridging and trading systems.


PR:

 github.com/ethereum/ERCs



Add ERC: Cross Chain Intents 858

ethereum:master ← marktoda:add-cross-chain-standard-erc

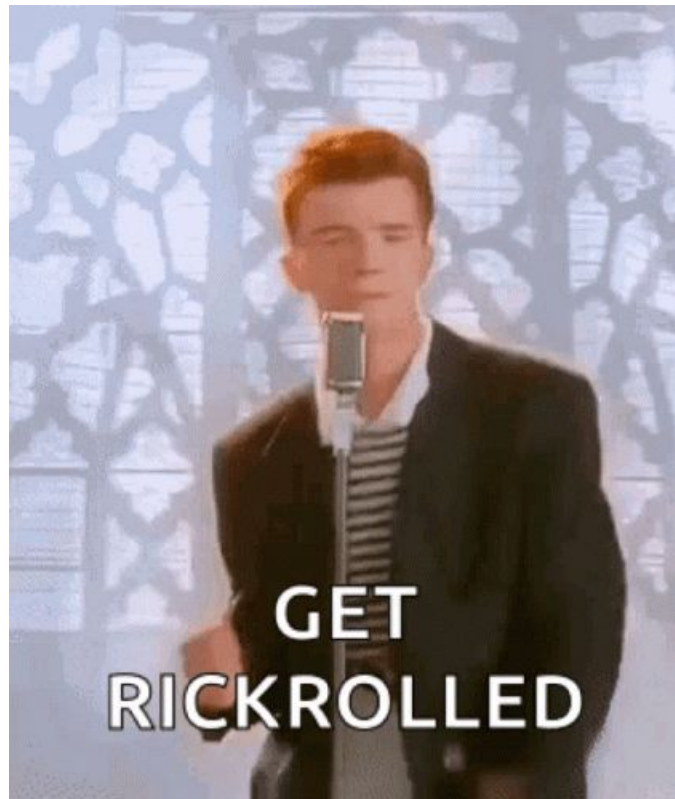
opened Apr 11, 2024  marktoda +178 -0

This commit adds an ERC for a Cross Chain Intents Standard which is meant to uni...

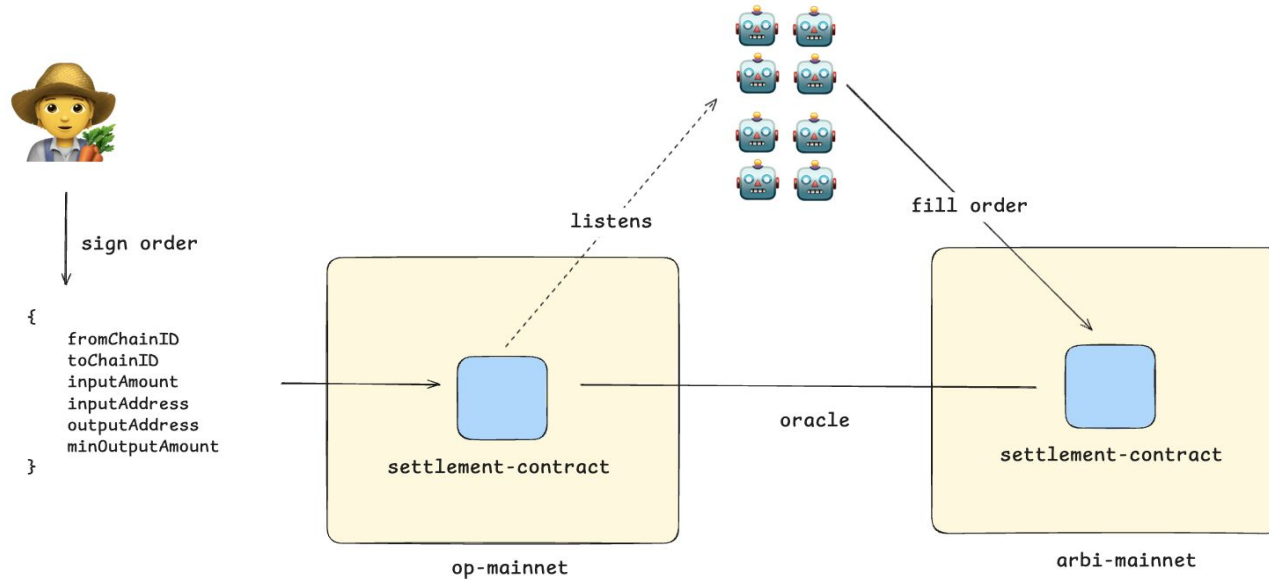


What are intents?

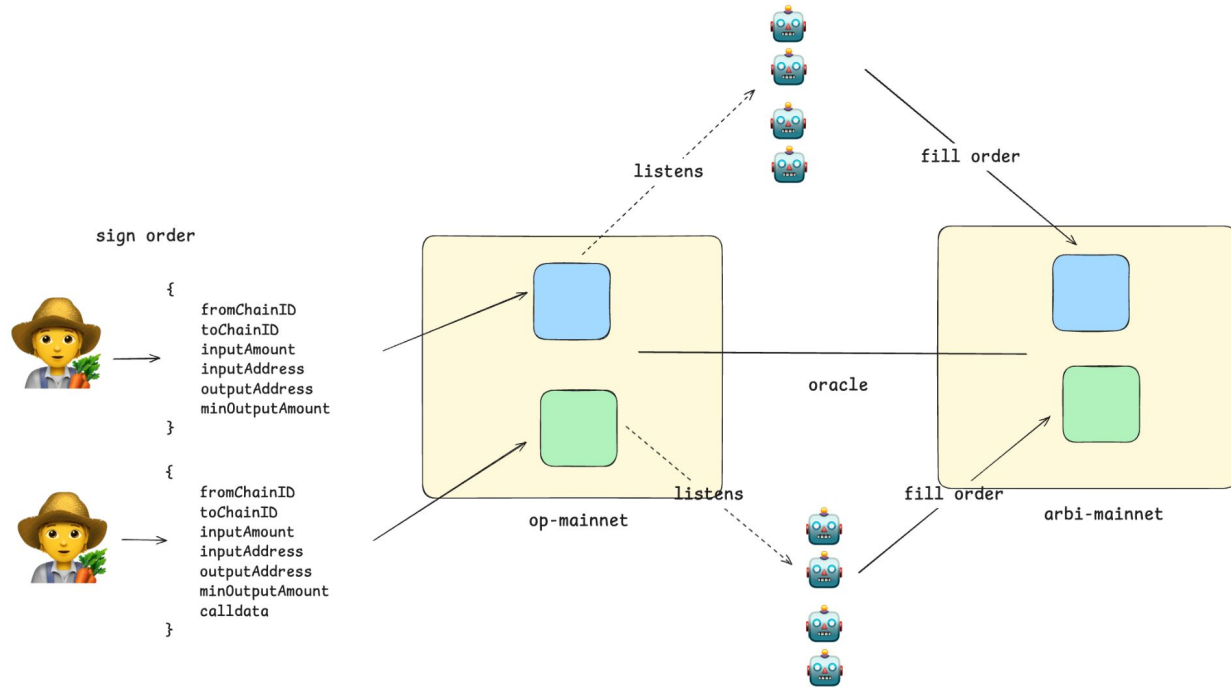
What are intents?



Goals of the ERC – Eliminate Filler Fragmentation



Goals of the ERC – Eliminate Filler Fragmentation



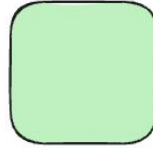
Fragmentation is bad for Everyone

USER



- loss of efficiency coz lack of competition
- doesnt understand what they are signing

Intent-Protocols



- Becomes expensive to attract fillers

Fillers



- Access to orderflow requires more work

Ideal World

- New intent protocol launches
- Fillers start participating in both at 0 effort

How does a filler decide to fill

- Can I interpret?
 - Do I understand the order type and other necessary interfacing?
 - Do I understand the protocol the order is trying to use to execute?

Users and fillers first need a shared language to talk and communicate their needs/demands to each other

How does a filler decide to fill

- Can I interpret?
 - Do I understand the order type and other necessary interfacing?
 - Do I understand the protocol the order is trying to use to execute?
- Can I fulfill?
 - Can I satisfy the asks made i.e provide enough output tokens, use the oracle user wants to use for settlement?

Now that I(filler) understand the user request, do I accept the terms and conditions

How does a filler decide to fill

- Can I interpret?
 - Do I understand the order type and other necessary interfacing?
 - Do I understand the protocol the order is trying to use to execute?
- Can I fulfill?
 - Can I satisfy the asks made i.e provide enough output tokens, use the oracle user wants to use for settlement?
- Do I want to fulfill?
 - Does it make economic sense for me to fulfill aka are profits meaningful enough to bear the risk?

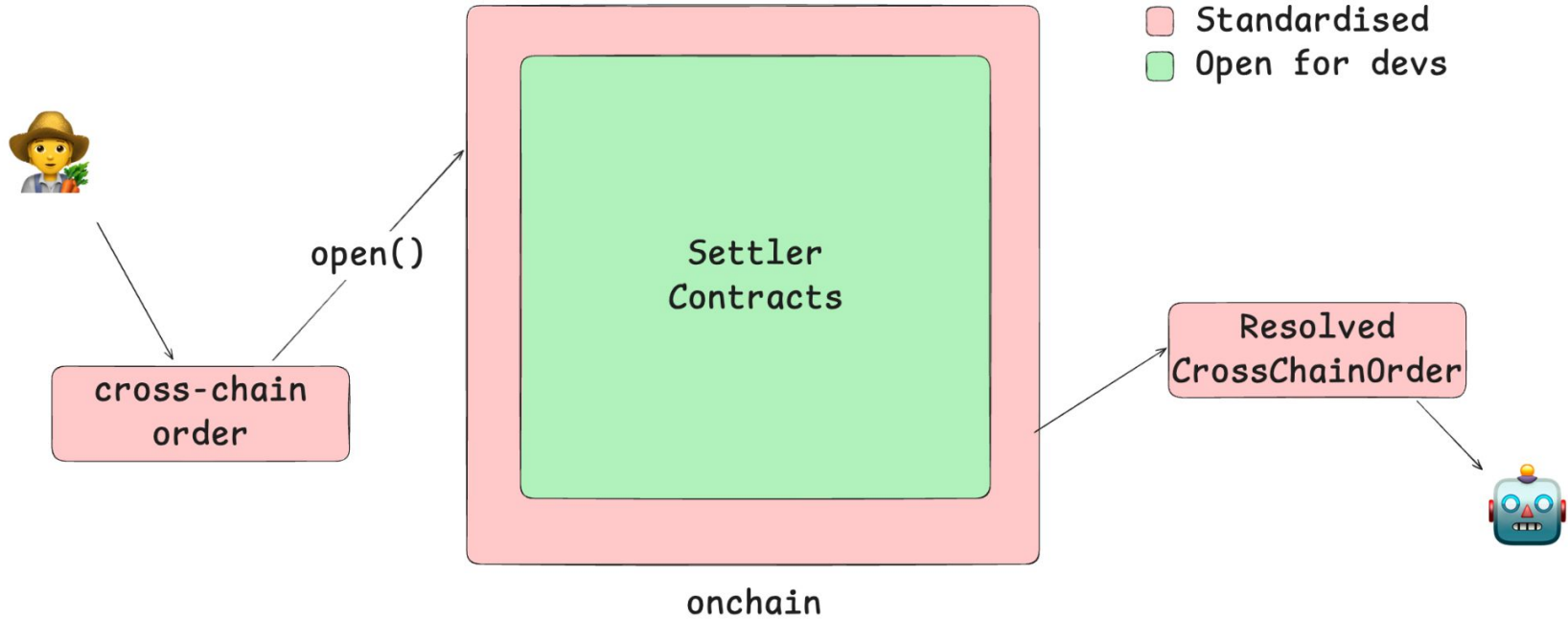
Is there enough on the table here for me for it to make economic sense?

Why does filler fragmentation exist?

- Every order/intent-protocol is basically a shared language between user and filler
- How many languages do you know?
- These are financial contracts and “misunderstandings” in communication can cost serious \$
- Fillers spend most time making sure they understand the language

Manual Interpretation is fundamentally *not* scalable. We need google translate

Quick 7683 Explainer – Settler Contracts are google translate



What's inside settlement contract



Filler POV – 7683 compat IntentProtocol#1



Filler POV – 7683 compat IntentProtocol#2



Core Issue: Fillers can't trust data from unauthenticated contracts

- Since devs can play/poison the settler contract, fillers can't accept standardised-resolved-orders as input to their systems
- Because the settler contract aka the translator can simply lie about the user-intent
- Filler systems need to take in the entire settler-contract + standardised-resolved-order as input in their decision making tree

Making it again, hard to have the same fillers operate across multiple protocols even if they use the same 7683 standard

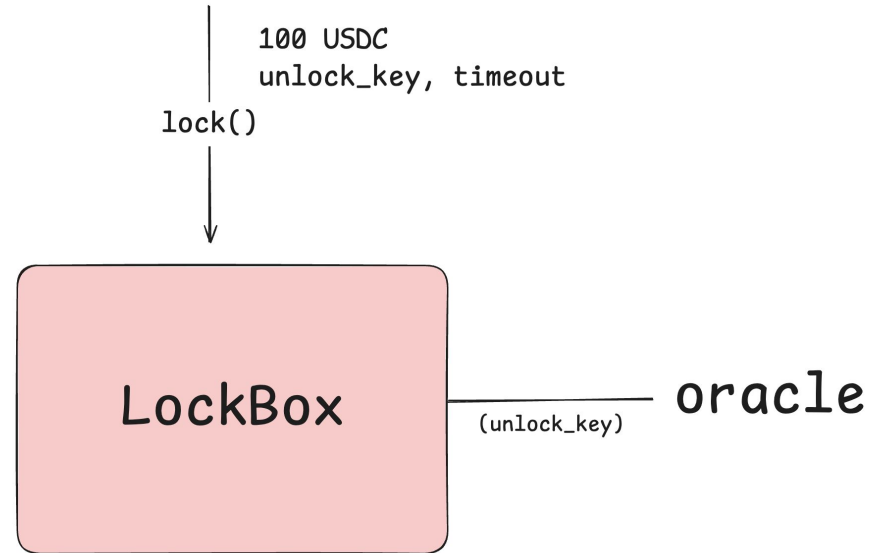
Potential Solutions

- We need to start providing assurances both to users and fillers
- Having a giant black box ie the settler provides no assurances forcing users and fillers to research/audit every settler they interact with
- Solution?

Break down the giant “settler” contract into smaller credible primitives that provide guarantees

Solutions – Standard Escrow Implementations

- Standardised lockbox so both users and fillers now have strong assurances
- Assurance that funds exist inside the lockbox till
 - Either the oracle
 - Or user claims (post timeout)
- Now no matter what the settler contract does, as long as funds land on the lockbox you don't have to care how it operates
- You can go to fill() on destination and verify the payload oracle passes through to do key validation



END

- Intents are amazing, need to end fragmentation of fillers
- 7683 while is a step in that direction, doesnt take us there **fully**
- A bunch of additional modules on top can take us there, hit me up if you want to collaborate on figuring out what exactly.