The general procedure:

图示

描述已自动生成

Event-process.py

This python file is used to decode event/logs of a tx. It includes one class named Event\_checker. It responsible for providing required ABI files.

Web3\_provider.py

This file includes many auxiliary functions and classes, which are used to provide web3 connection, contract factory, ABI file and well divided ABI sets.

Class W3:

class used for initialization period, responsible for providing web3 connection, target ABI file and contract factory.

Class Reader:

The class used for extract specified ABI file if it has been well processed and downloaded. It will assort function ABI and event ABI into two different list. The return type is a dict contains event ABI, function ABI, contract name and entire ABI files.

Function combine\_tx(contract hash)

In fact, this function is the aggregation of the entire file. Most of times, you can just call it if you want to get a ABI to decode some info.

Beyond picture will show how it works:

**图示

描述已自动生成**

New\_Abi\_checker.py

The pivot of entire program, its workflow has been showed in page one.

Problem and flaws of program:

1. Possible redundancy of format transform

Because we want to write processed result into csv file, the Json format must be changed due to the complex indent, which means we need to grab out every item in list value or dict value and pave them in plain to the dict.

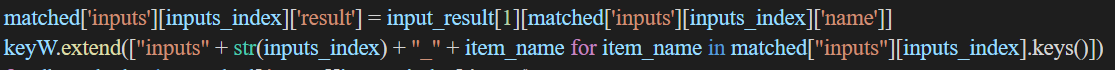
My method is in every iteration of such value (e.g inputs in tx and receipt), we will create a new element named [“inputs” + str(index) + “\_” + key] and insert them in the last of dict. And in the last, we will delete original indented “inputs” to eliminate indent.

Before: After:

文本

描述已自动生成

Problem is codes in here is a bit complicated and verbose, for example:



You can see in there if we want to access a value, we need to add 3 suffix! And out of this statement already has two iterations.

Is there any method that can avoid adding so many suffixes?

2. increase process speed by multiprocessing and concurrency

Ideally, in the last the entire workflow will be like:

图示

描述已自动生成

3. failure to match ABI file and event topic according to given contract address

In a transaction which hash is

0x8a7a6774e8cde45975941f5dd3151439b272f66900b210f9700ec5fbac3aa7db

One of its events named “minted”, usually we get ABI file according to attached contract address in event. But for this one we can’t find its corresponding neither in the given contract address nor local ABI database.

In case of such problem happen in future, we use log to record it once we detect such thing happen.

Next stage:

After we finished this stage, we are going to focus on trace.

In general, the known method to parse a transaction is:

decode input/event and trace. So, for our assumption, in the last this is what we can do:

图示

描述已自动生成

For our final assumption, we can build a system which support retrieve txs under specified address, detailed parser a tx process and workflow analyze. Its kind like combine tenderly, Etherscan and dune into one system. And with good compatibility, it can provide users with functions similar to the combination of tenderly and etherscan, etherscan and dune at any time, with high data provide/retrieve flexibility.