If you don’t know where to start, getting all the transactions of a smart contract address

can be cumbersome and time-consuming. In this tutorial, you will learn several techniques for querying and retrieving all transactions from a contract address.

We’ll look into getting smart contracts transactions manually, using centralized services such as Etherscan or Dune Analytics, and even building your own solutions using popular SDKs and libraries powered by companies like Alchemy (Alchemy SDK), and Quicknode.

If you're new to all of this, checkout oursolidity smart contract development course on Cyfrin Updraft, completely for free.

Let’s get started!

How to fetch all the transactions from a smart contract address

Querying a node directly for all transactions in a smart contract address is difficult due to the way the blockchain stores data but a key skill to learn for web3 developers. Blockchains like Ethereum aren’t relational databases themselves and are not designed for easy querying. They are a string of blocks connected to each other, a sort of linked list.

Because of this, if all you have is a node, you would need to “brute force” go through each block, gather a list of all transactions, and check if the transaction originates or is sent to the address you are looking for. Additionally, you’d need an archive node - an instance of an Ethereum client configured to build an archive of all historical states.

Luckily though, lots of services out there have already built optimized solutions storing and indexing these transactions in databases ready for you to be queried.

We will look at each example to query transactions from a smart contract address as well as get notified when a new transaction happens.

We’ve put together a GitHub repo with all the examples for you to explore!

Why retrieving all the transactions from a contract address is important?

There are cases where smart contract auditors and security researchers need this information to learn more about protocols:

- Does a protocol have a history of silently upgrading features without telling the community?
- How can you see what contracts are connected to each other?
- Want to find exactly when an exploit was executed?

Others where web3 dapps developers need to show this information to their users, or where DeX need this information to perform certain actions on their platforms.

Regardless of the use case, fetching historical and current transactions from a Smart Contract address is helpful whenever you’re dealing with Blockchain data.

Let’s get started!

Get started fetching all the transactions from a smart contract address

For our example, we will use theEthereum Mainnet Aave token, as it’s a proxy contract and we can also show some event querying aspects like “how many times it’s been upgraded by governance.”

First, let’s learn how we would fetch blockchain transactions to and from a smart contract if we were to do it manually - brute force.

1. Brute Force - Query the blockchain to get a smart contract address transactions

As said before, fetching Smart Contract transactions directly querying the blockchain is definitely the most time-consuming of all the solutions we’ll explore in this article - but also the most eye-opening when it comes to understanding how transactions work.

Here is an example of brute force getting transactions withweb3.py and one using ethers.js - two of the most popular web3 libraries to date.

Let's start by adding a start block and end block, to avoid making a bazillion calls to your node.

With web3.py we can do the same:

As we can see above, the "brute force" method involves going block-by-block transaction-by-transaction until you find all the events you're looking for, which can take a very long time.

And here is the worst part this list is non-exhaustive

! If your address is a contract, contracts can make transactions to and from each other in a single transaction but will not be the main "to" or "from". Let's see why:

Contract-to-contract getting smart contract transactions

First, let's quickly refresh how the transaction object will look like

As we can see we have a "From" and a "To" fields containing the address where the Tx originates, and where it ends - we're interested in every transaction that contains our contract address in the To or From fields.

Here's the issue:

Let's say that a Wallet interacts with a contract's function "doStuff()", we'll call it contract A. In that case, the "from" field will be the Wallet's address, and the "To" field will be our contract address - awesome!

Now, let's say that in the doStuff() function, the contract will be interacting with another contract, called "contract B" - in that case "from" and "to" will still just be A & B even though we interacted with C.

The code above would miss this contract-to-contract call! And we would miss a transaction.

To get all of those transactions, you'd have to simulate each transaction running something like [debug_traceTransaction

](https://docs.alchemy.com/reference/debug-tracetransaction) keep track of the stack, and keep track of which opcodes interact with other contracts.

It's a pretty laborious and time-intensive process and, ultimately, using the code above would only give us a subset of all the Smart contract transactions.

You could write all this, but most op to go with indexing transactions into an easier-to-query database or use a service that makes it more accessible.

Get all the transactions from a contract address - Centralised Services

Luckily there are services out there that are making our job of retrieving all transactions in a smart contract or address way easier. Let's explore them, and understand the trade-offs.

An example of internal transactions from etherscan

One of the easiest ways to get a list of transactions (and internal transactions) is to use a block explorer likeetherscan.

For our Aave token, seeing a list of transactions is very easy. If we select internal

and toggle on Advanced

we can see a list of contracts that have interacted with Etherscan.

On Etherscan's back end, they are indexing all this data for us to "see" it more easily.

They also have an advanced filtered mode, but unfortunately, we would still lack some granularity, and getting exactly the transaction info we want can be a little tricky.

Let's see how Dune Analytics, partly, solves this.

2. Get a smart contract transactions using Dune Analytics

Dune Analytics is a platform that allows users to create, share, and explore Ethereum data dashboards. It provides insights into by allowing users to query and visualize data in a user-friendly manner.

Simply put, it indexes blockchain “stuff” so that you can query it as if it’s a SQL database. For example, we can easily see all transactions in our Aave Token.

First, if you haven’t already, let’s navigate to Dune analytics and create a new account.

Then, navigate to the “Queries” tab and click on the “New query” button.

This will open up a console-like editor, where you’ll be able to write SQL queries in it.

Copy and paste the following code in the text editor and click “Run”:

After a few seconds, you should see all the transactions appearing in the table below - more than 345,000!

Dune comes with a table known as ethereum.traces, which includes all the transactions in a contract. The query above will find EVERY transaction that interacted with the Aave token or the Aave token interacted. Saving you the hurdle of having to simulate every transaction as we had to before with our Brute force method! ADD ANCHOR

Now, this is great if you want a quick way to visualize all the transactions from and to a Smart Contract or wallet address. You can also implement it in your code using the Dune API - but what if we want even more granularity and flexibility?

In that case, we might want to use something like the Alchemy SDK or the Quicknode SDK

4. Alchemy

In this case, we’re going to use the popular Alchemy SDK - which offers a set of methods and APIS to allow us to fetch Blockchain data with ease.

One of these is the Alchemy Transfer API that allows you to easily fetch historical transactions for any address across Ethereum and supported L2s including Polygon, Arbitrum, and Optimism.

The getAssetTransfers method does in fact something similar to what we’ve done before using Dune Analytics - Plus it will give you some extra granularity over the data you want to fetch - like External transactions vs. Internal, or filter them by contract standard.

To get all transactions from and to our Aave token smart contract you can copy the following code:

This will print a list of ALL the transactions from Block 0 to the latest available block - in a few milliseconds!

BTW, For this, you can also use the Python SDK Patrick Collins made to make it easier to work with the Alchemy API in Python, here’s an example:

5. QuickNode SDK

QuickNode SDK - offers various modules, each having methods that make fetching blockchain data easier than getting the raw data and having to filter them manually.

One of the modules, called the API module, lets you make GraphQL queries leveraging the QuickNode Graph API to make intended queries and get only the valuable data. Right now, it is supported on Ethereum and Polygon, while core RPC functions in SDK are supported on all EVM chains.

The events.getByContract query returns the events for a smart contract provided the contract address; the query can be formatted to filter transactions based on block number, wallet address, type, etc. The response is paginated, making it easier to navigate larger payloads.

To get all transactions from and to our Aave token smart contract, you can copy the following code:

This will print a list of ALL the transactions from Block number greater than or equal to 1 to the latest available block (in a paginated format) - in a few milliseconds!

Congratulations! You've just learned how to get all the transactions from a contract address!

Conclusion

In this guide on how to get all the transactions from a smart contract or wallet address, you've learned everything you need to know to retrieve the information you need from any on-chain activity. If you want to get more in depth and learn web3 development, make sure to checkout our smart contract development and auditing courses on Cyfrin Updraft, completely for free!