Listening to On-chain Events on Solana

Introduction

One of the most fundamental parts of being a developer in crypto is setting up systems that listen to the blockchain. You might be listening for payment confirmations, NFT sales, fund transfers, or you might just be monitoring accounts for security purposes. Whatever your use case is, it is crucial that the systems you build for watching the chain are fault-tolerant, reliable, and optimized for latency.

In this post, we'll go over a few ways of building such systems on Solana, example use cases, and how you can build better systems going forward with Helius.

Listening to the Blockchain

There are two main ways of listening to on-chain events:

1. Polling
2. Streaming

Polling is a method where the client or application repeatedly checks a server or data source for new data. This can be done at set intervals or on-demand. When a request is sent to the server, the server responds with the latest data, whether or not there has been any change. This means that the client or application may receive the same data repeatedly, even if there has been no update.

Streaming, on the other hand, is a technique where the server pushes data to the client or application when there is an update. This means the client does not need to request data repeatedly, and the server only sends data when there is a change. This results in a more efficient and real-time data transfer, as the server sends only relevant data.

While polling is a simple and widely used technique, it can result in high network traffic and can be resource-intensive, especially when frequent requests are made. Streaming, on the other hand, is more efficient, as the server only sends updates, reducing the amount of data transferred and network traffic.

Streaming is typically used in applications where real-time data updates are critical, such as stock market feeds, social media updates, and online gaming. Polling is more commonly used in applications where data updates are less frequent or critical, such as email clients or news websites.

While both techniques have their advantages and disadvantages, streaming is generally a more efficient and preferred method for real-time data updates, while polling remains useful for certain applications where data updates are less frequent.

For most types of workflows on blockchains — especially Solana — you'll want to stick with Streaming. There are a few reasons for this but Streaming will be much simpler to implement and will be easier to handle for high-traffic use cases, which Solana specializes in.

How to Listen for On-chain Events on Solana

Solana is a very fast blockchain. It emits new blocks every 400ms and these blocks generally contains thousands of transactions. The crazy part is that this is the slowest it will ever be. As the core engineers keep working on the system, the rate of new data being processed by the chain is just going to increase. As a result, it is crucial that you design your systems with scalability in mind.

Luckily, you have a few methods for listening to on-chain data on Solana:

1. Polling
2. Websockets
3. Geyser
4. Helius Webhooks

Polling

This is the least-preferential way of listening to events on Solana, though it may conceptually be the simplest.

Depending on what data you're trying to get, you would simply set up a loop which repeatedly calls a JSON-RPC method on a Solana RPC (like Helius). Common use cases here would be polling the `getBlock` method for listening to new blocks or

polling `getSignaturesForAddress` for checking up on new transactions for a given address.

Interestingly, this might also be the best method for very advanced use cases where you have very custom logic for triggering new updates.

Websockets

Solana RPCs also expose PubSub websockets for developers to hook into. The available event types are:

- accountUnsubscribe
- accountSubscribe
- logsSubscribe
- logsUnsubscribe
- programSubscribe
- programUnsubscribe
- signatureSubscribe
- signatureUnsubscribe
- slotSubscribe
- slotUnsubscribe

Here's a small code-sample for listening to account changes via a Helius RPC (which supports websockets), in JavaScript:

One very important caveat with websockets is that while they're very useful for prototyping, we've found them to be quite brittle and unreliable in practice. It is very strongly recommended that you do not use them for mission-critical workflows as you will miss events.

Geyser

TL;DR — you can have nodes on Solana stream data to you directly through a plugin interface that customizes how you receive that data. This is the fastest and lowest-latency way of streaming data on Solana and is an absolute must for things like DeFi liquidations and latency sensitive applications.

Solana Validators and RPCs have a unique, Solana-native way of streaming data: Geyser Plugins.

The validators have been enhanced to support a plugin mechanism, called a "Geyser" plugin, through which the information about accounts, slots, blocks, and transactions can be transmitted to external data stores such as relational databases, NoSQL databases or Kafka. RPC services then can be developed to consume data from these external data stores with the possibility of more flexible and targeted optimizations such as caching and indexing.

Unfortunately, setting up Geyser can be quite involved and expensive. Even after you've set it up, you'll need a good amount of DevOps work to make sure it's running smoothly. Luckily, at Helius we've developed something called GeyserVM — which lets you upload your plugins and have them running in mere seconds. Our highly available and redundant clusters ensure that you don't have to worry about any data issues and since the resources are shared, you're able to save over 200% on your monthly costs!

If you would like dedicated Geyser-as-a-Service, we offer that as well of course.Click here to learn more about it.

Webhooks

Webhooks are a simple concept. Simply put, webhooks listen to events and send them to a server you've setup when they happen. For example, you might be interested in sending your Discord server a notification whenever an NFT sale happens — a webhook can accomplish this seamlessly.

For most event-listening workflows on Solana, webhooks will be the easiest, most flexible, and most cost-efficient method. The only time webhooks should give you pause is for extremely low latency use cases where the difference in 5ms might make or break your application — like high frequency trading.

At Helius, we've built the most robust webhook offering in all of crypto.

You can listen up to 100,000 addresses (per one webhook!), configure the types of events you want to listen to (we parse

these for you!), and simply put in the URL of your server — that's all you need to get started, for free.

One of the many benefits of a service like this is that not only do you get to save weeks or months worth of developer time and effort, but you also get an elastic backend that scales with you as you grow.

To read more about webhooks, click here.

Use Cases

Okay, so now you know how to listen to on-chain events on Solana, what should you do next? Here are a few use-cases that might be interesting:

Bots

- When an NFT is listed on marketplace X, trigger an "nft buy" action.
- When a margin position is unhealthy, trigger a "liquidation" action.

Monitoring & Alerts

- When a program emits a certain log, trigger PagerDuty integration.
- When a token account balance changes by more than X%, use Dialect to communicate a warning action.

Event-driven Indexing

- When any transaction occurs for a given program, send it directly to your database or backend.

Notifications & Activity Tracking

- When a transfer occurs from wallet X to wallet Y — send a Slack notification or email.

Analytics & Logs

- When event X happens, send it to an ETL pipeline or persist it directly on Helius to view trends over time.

Workflow Automation

- When event X happens, trigger any set of actions.

Conclusion

In this post, we've covered the differences between polling and streaming, different ways of listening to the Solana blockchain, and some example use cases.

In conclusion, you can listen to on-chain events on Solana via polling, websockets, Geyser, or Helius webhooks. Each have their pros and cons and it's important that you consider the requirements of your systems before deciding which approach to implement.Thank you for reading!

To learn more about Helius' offerings including Solana RPCs, Solana APIs, Solana Webhooks, and Solana Infrastructure: please visit our website.