# Chaintool

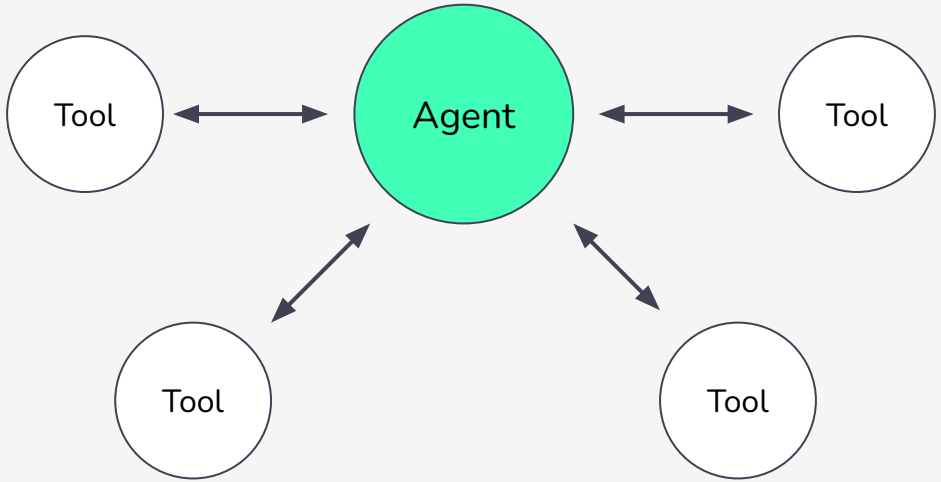
*On-chain toolchain for dynamic & agentic tools.*

Erhan Tezcan
https://github.com/erhant/chaintool

February 9, 2025

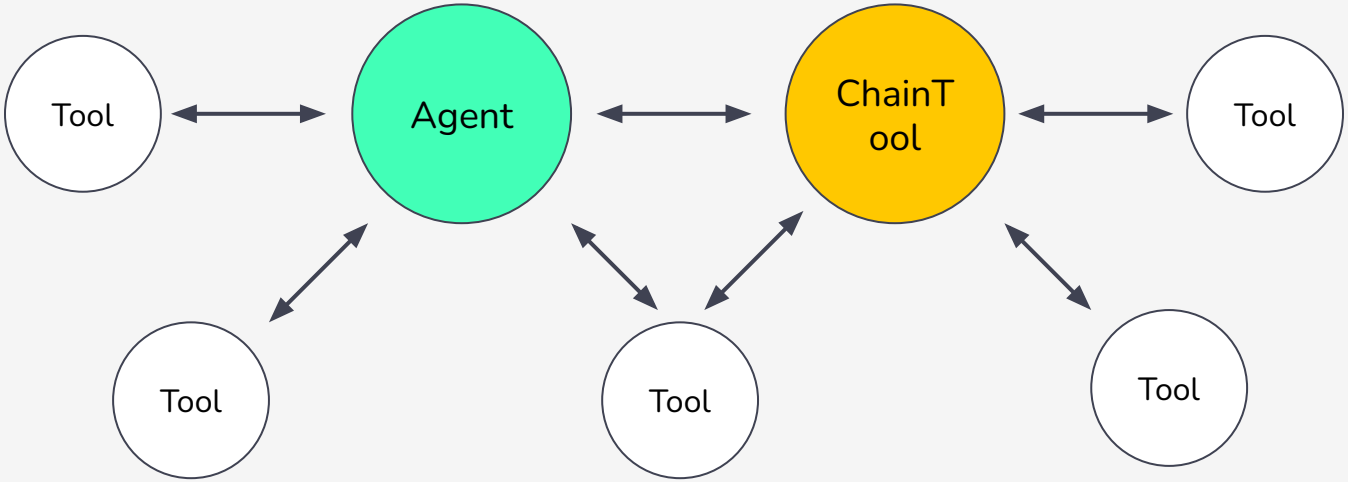
# Agents & Tool-Calling

Midst of the AI Agents on-chain, we see that all tooling for the agents are defined within the agent code itself. Agentic frameworks like LangChain, ElizaOS, AgentKit etc. all define extended tool-calling capabilities with hand-written plugins and actions.



# Chaintool

**Chaintool** allows for dynamic tool-calling capabilities for tools designed to interact the with blockchains. It defines tools as instantiations of a struct on a Smart Contract, and provides the tools to interact with them.



# Chaintool: Definition

In a usual agentic tool / action, we have 4 things that define our tool:

- **name**: LLM-friendly name of the tool
- **description**: LLM-friendly description of the tool
- **input schema**: inputs for the function
- **invocation**: the actual code, returns a string.

In a **Chaintool**, we again have 4 things that define it:

- **name**: LLM-friendly name of the tool
- **description**: LLM-friendly description of the tool
- **abitypes**: human-readable ABI types for the contract functions
- **target**: the contract address as the tool target

# Chaintool: Definition

For example, lets say we have an `AddTool` contract at some address `0xdead..beef`, we can instantiate it with the following Chaintool:

- **name**: Add
- **description**: Adds two numbers together and returns their sum.
- **abitypes**:
  `["function add(int256 a, int256 b) pure returns (int256)"]`
- **target**: `0xdead..beef`

# Chaintool: Definition

The functions listed within abitypes are human-readable (thanks to https://abitype.dev/) which helps the LLM to understand them & prepare arguments.

- If a function has `view` or `pure` state modifier, the agent makes a `call`.
- Otherwise (`non-payable` or `payable`), the agent makes a `send` transaction. It can send with value as well!

# Chaintool: Registry

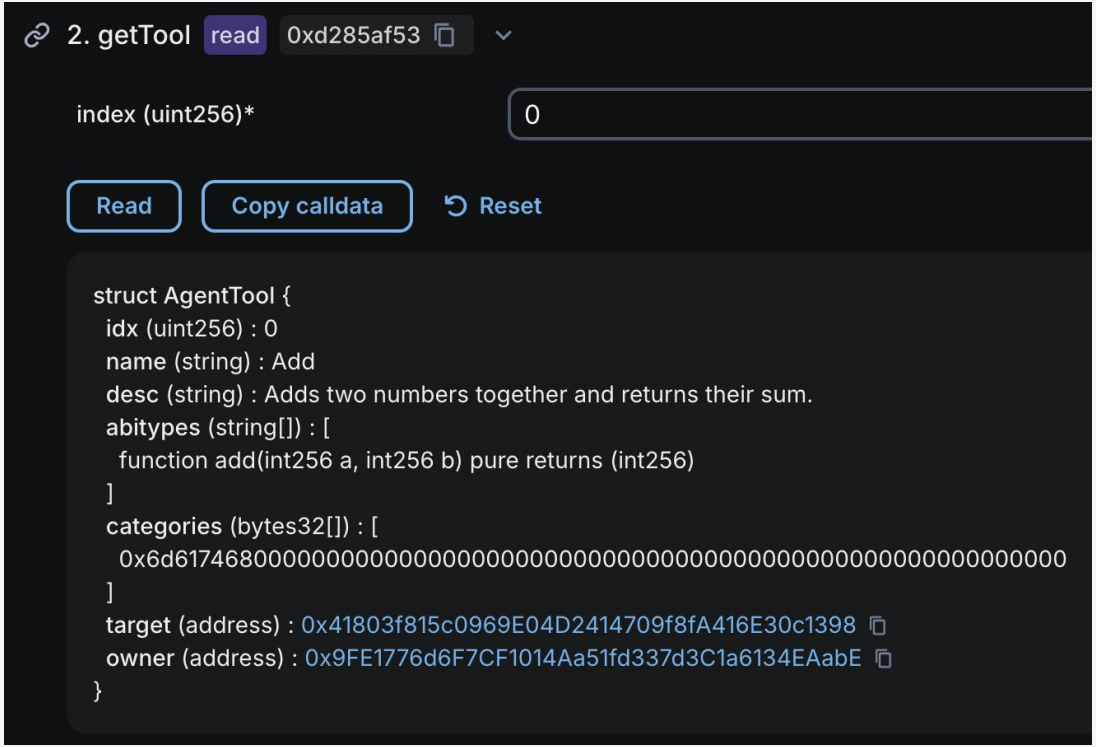
**So how does an agent know which Chaintool to use at all?**

The main discovery point of Chaintools is a Chaintool registry contract. It is where Chaintools are registered & the details of which are accessed from.

Every tool in this registry are specified along with **categories** that the agent can filter through, e.g. a token Chaintool could have `defi` and `erc20` categories; or a Trigonometry Chaintool can be in `math` category.

Each registered tool has a tool index given to them when they are first registered to the contract.

# Chaintool: Registry



# Chaintool: Actions

With this project, we provide the following actions to the CDP AgentKit:

- **getChaintoolsCategories**: reads all available categories
- **getChaintoolsByCategory**: reads the registry to get Chaintools, optionally with category filters.
- **getChaintoolByIndexAction**: after selecting a Chaintool, it can get its details (e.g. abitypes) via its index
- **useChaintoolAction**: it can invoke a Chaintool after reading its details with the actions above, it provides the `abitype`, `target`, `toolArgs` and an optional ether `value` to invoke this action.

# Chaintool: Actions

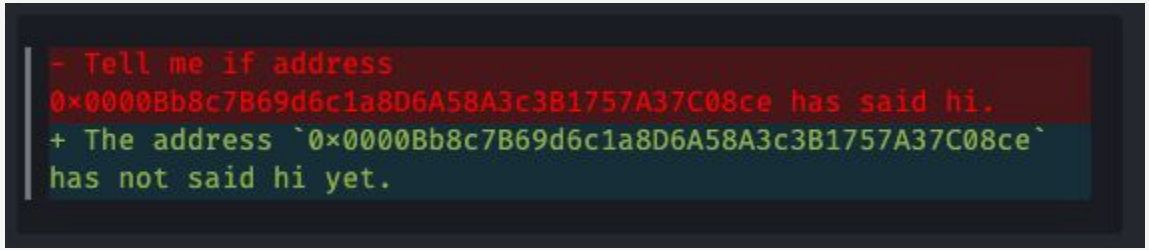
We can prompt the agent to fetch all Chaintools from the registry.

# Chaintool: Actions

We can ask for details of a specific Chaintool.

# Chaintool: Actions

With the information from previous prompts, the agent has all the information (abitypes, target address etc.) to make a Chaintool call!



# Chaintool: Actions

Note that the Agent's Wallet is based on CDP Wallet Provider. The seed of a CDP wallet export constitutes a master seed for a Viem HDKey account!

So, upon first initialization, we also create an HDKey account in Viem to provide a Viem Wallet client with public actions as well.

We use both of these two interact with our tools, the Viem is mostly used for Chaintool interactions.

# Chaintool: Actions

We also provide a **Nillion SecretVault** integration to the agent, where it can store a history of Chaintools that it has used so far, and read the vault to remember them at times. The stored data are encrypted via Nillion!

This is done via a pre-defined Nillion schema, along with 2 actions:

- **nillionRead**
- **nillionWrite**

# Chaintool: Agent

```
// initialize AgentKit
const agentkit = await AgentKit.from({
  walletProvider: cdpWalletProvider,
  actionProviders: [
    // no ERC20 or WETH here :)
    pythActionProvider(),
    walletActionProvider(),
    cdpApiActionProvider({
      apiKeyName: cdpConfig.apiKeyName,
      apiKeyPrivateKey: cdpConfig.apiKeyPrivateKey,
    }),
    cdpWalletActionProvider({
      apiKeyName: cdpConfig.apiKeyName,
      apiKeyPrivateKey: cdpConfig.apiKeyPrivateKey,
    }),
    // chaintool
    getChaintoolsByCategory(registryAddress, viemClient),
    getChaintoolByIndexAction(registryAddress, viemClient),
    useChaintoolAction(viemClient),
    // nillion
    nillionRead(nillion.config, nillion.schemaId),
    nillionWrite(nillion.config, nillion.schemaId),
  ],
});
```

# Chaintool: Demo Time!

Demo time!

- We will show our Chaintool viewer online at https://chaintool.vercel.app
- We will show an CDP AgentKit using Chaintool & Nillion

# Chaintool: Future Work

**(1)** The registry contract of Chaintool currently has no authorization logic, nor a validation logic. We would like to make it more hard & worth-it to register a Chaintool.

**(2)** Following (1), the tools become more valuable for everyone; so the next sensible thing is to create an economy around them & value their creators! AI Agents already have an economy (e.g. Virtuals) but Chaintool paves the way for Agentic Tools to have an economy as well.

**(3)** We would like to construct more advanced tools with more impact for the agent as well, paired with an agentic profile that is stored on-chain to keep track of its tools from its own contract as well.

# Chaintool: Future Work

**(4)** Implement Chaintool plugins & actions to existing frameworks, especially Eliza. Upon inspection we realize that Gelato Relay comes close to Chaintool's purpose as well, so an integration with Gelato Relay plugin is also considerable.

# Fin

https://github.com/erhant/chaintool

**x:** 0xerhant | **tg:** tgerhant