Introducing Truffle DB, part 2 - 'Weight and Switch'

Welcome back! If you missed thefirst part of this post, you may want to read it to learn more about Truffle's contract artifacts, their current limitations, and why we seek to build a better solution.

Or, just to catch up: * Truffle currently stores information about your smart contracts in files that we call "contract artifacts" (or just "artifacts" for short.) * Truffle uses these artifacts for pretty much everything. * Plenty of other tools use artifacts too, and rely on their behavior. * Unfortunately, the artifacts file format has several design flaws, forcing dozens of workarounds and even preventing certain use cases entirely. * Addressing these flaws head-on risks severe negative impact on other projects. * We're excited to share a project we've been working on, to offer a deprecation plan and to encourage community involvement.

So why should you care? * If you are a tools developer, or rely on Truffle artifacts directly in any way: there will be breaking changes! * If you use Truffle for smart contract development: the developer experience using Truffle will have fewer roadblocks ("software warts "). This will not break existing Truffle projects. * No matter who you are: better data means better tools.

Great! I'm convinced. 😁 Now, where were we?

What is Truffle DB?¶

Truffle DB comprises two key ideas, currently at different stages of development. These are:

1. A data model for organizing your project's smart contracts: how to keep track of all the different pieces of information, from your contract source files to where they are deployed.
2. A flexible JSON data access interface: how to read exactly the data you need and how to update the DB with the data you have.

Data model¶

Data model excerpt: how contracts, sources, compilers, and compilations all relate. (Text-accessible version available via diagram source) What is a smart contract? What is your contract's bytecode, what is its source code, and how do they relate? By looking at how Truffle expects these relationships in practice, and considering how these concepts are commonly understood, we hope to make life easy by creating a shared vocabulary for interacting with the smart contract domain.

Diagrams are worth more than words here, so please check out our data model docs to get a broader sense for what this represents.

Current status : Implemented as minimally viable, designed to be extended.

Flexible interface¶

Concept image: Example query to fetch specific project contract info. (See gist for image text) Truffle DB usesGraphQL for its primary interface. GraphQL affords flexibility and a data-first design methodology that suits our goals well. This interface is intended for use in tools to build on top of Truffle; as a normal user you won't have to touch this. (Of course, we hope this will encourage you to find new ways to solve problems by building and sharing tools of your own!)

Truffle DB is still in development, but we would like to share what we've got so far and to talk about where we're looking to go. Although it's early, we want to increase awareness about future breaking changes, and we hope to paint a picture of some new foundations that we're excited will help the smart contract development tools ecosystem.

Current status : Internal infrastructure in place, prototyping for public review to begin soon.

What we're releasing first¶

We're gearing up for a minor release (Truffle v5.1), and we plan to make a first version of Truffle DB available as an experimental implementation. This means building blocks to start: key pieces of infrastructure to paint a picture of Truffle DB for review and iteration.

Day 1, this means you will get: * The ability to convert your artifacts and load them into Truffle DB * A mechanism to persist Truffle DB data side-by-side with your artifacts * Improvements to Truffle that take advantage of this data when it exists * A bundled GraphQL playground for hands-on exploration * The ability to easily explore the schema to better understand our system design

Interact with Truffle DB: Apollo's GraphQL Playground provides a handy way to work with GraphQL interfaces. We're bundling a Playground instance into Truffle DB to make it easier to learn about. (For text-accessible version, see query source, query variables, and the query result)

What's left to be done (soon)¶

Since we've started by building infrastructure, the work remains to connect the dots into a final design that we can validate for quality and usefulness.

This includes: * Documentation, of course! See our current Truffle DB – Proposal working docs, which include the data model diagrams. Expect us to convert these docs into proper documentation for release. * Better persistence. Our initial implementation will save JSON files to disk, but we intend to support more robust data storage options, including relational databases and native support in Truffle Teams . * Schema design updates, including support for pagination and proper data filtering. * A prototype candidate for Truffle DB's external interface.

The long-term plan¶

We're going to introduce Truffle DB gradually, deprecating the old artifacts format and introducing Truffle DB more and more as it's validated.

We expect to follow these steps: 1. Release initial Truffle DB features as opt-in and experimental. 1. Gather feedback and support to build a cohesive Truffle DB release candidate (for broad community use). 1. Turn Truffle DB on by default, but continue to use artifacts as the primary data store. 1. Convert the rest of Truffle to use Truffle DB as its primary data store, but continue to maintain artifacts alongside it for compatibility. 1. Continue to maintain both data stores, and switch to using Truffle DB to manage artifacts (instead of existing artifacts management code). 1. Turn off artifacts by default, but maintain legacy support through configuration. 1. Eventually, remove artifacts altogether.

We expect this process, start to finish, to span at least two major (breaking change) releases. We plan to maintain legacy support for artifacts for at least 18 months after we release a Truffle DB release candidate.

via GIPHY

Weight and Switch: Dr. Jones demonstrating proper technique for delicate data migration procedures

What do you think?¶

Please reach out with questions/thoughts/use case ideas/implementation strategies, or even just to say hello. Your feedback will help us provide the most value and make sure we do so with enough clarity!

We're tracking overall progress in this GitHub issue (trufflesuite/truffle#1718) . Expect to see this filled out more in the coming weeks, so please follow along there if you're interested.

Thanks for reading! Hope to see you at TruffleCon!