NEAR CAST

NEAR Contract Analysis Subgraph Template

Author: Ryan Strawser

GitHub: r-strawser

r-strawser - Overview

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

https://github.com/r-strawser

Twitter: @RyanStrawser

JavaScript is not available.

https://twitter.com/RyanStrawser

Tech Specifications

Code

subgraph.yaml

purpose → meant to be easily ingestible by developers as it is the one of the two parts of the entire template they will be required to edit.

target audience → novice developers with no prior subgraph development experience (i.e. (1) dev who simply wants to see what functions are being called and returned in a contract, (2) dev who wants to see which functions are being called on a granular level so they can see the format of how the contract returns the data to be parsed. Then they are able to indicate which fields to extract from an entity (args, outcomeLogs, deposit, successValue, etc.) to build their mappings files and queries).

features → fullTextSearch: required to enable usage of the *functionSearch* fulltext search in the schema.subgraph file.

required user-configured fields - 3 fields →

dataSource

name: replace “<contract-name>” with the contract name (i.e. metapool).

account: replace “<contract-account-id>” with the contract account ID (i.e. “meta-pool.near”).

startBlock: replace “<block-height-of-contract-activity>” with the block height from the contracts last updated activity. This is found through by typing the contract account ID into the NEAR Explorer (i.e. the meta-pool.near contract lasted updated March 12 at block height #**61300633**).

schema.subgraph

purpose → entities that propogate the most useful transaction receipt information to developers. This should aid both developers and individuals inspecting the contracts activities in a searchable, organized fashion.

target audience → Developers who now have the hang of the template and want to start tailoring it to their own needs (i.e. now that Shirley understands the outcomeLogs for the mrbrownproject.near contract emits the “nft_transfer” event from three different function calls, she can optimize her subgraph to only check for those method calls).

Types ***subject to change with improvements***

Enums

```
enum EventStandard {
  NEP
  CUSTOM # events following no proposed format or specification
  NO_LOGGED_EVENT
}
```

Entities

```
type FunctionCallLog @entity {
  id: ID! # receipt_id

  # function call info
  methodName: String!
  executorId: String!
  args: String!
  deposit: String!
  outcomeLogs: String!
```

```
  statusValue: String

  # possible fields of outcomeLogs
  eventName: String
  eventStandard: EventStandard!

  # receipt info
  signerId: String!
  predecessorId: String!
  receiverId: String!

  # block info
  blockHash: String!
  blockHeight: String!
  blockTimestamp: String!
}
```

Schema

```
type _Schema_
  @fulltext(
    name: "functionSearch"
    language: en
    algorithm: rank
    include: [
      { entity: "FunctionCallLog",
       fields: [
         { name: "id" },
         { name: "methodName" },
         { name: "eventName" }
         ]
      }
    ]
  )
```

mappings.ts

purpose → populates the entities and focuses on readability. Proper documentation and commenting.

target audience → same as audience for schema.graphql.

Documentation

- **Quick Start**
  - Create subgraph instance in Hosted Service
  - Changes to make in the package.json so it is tailored to the devs Host Service instance
  - Yarn/npm commands to run
- **Usage**
  - Contract activity analysis
  - Subgraph Prototyping
    - Altering the schema, yaml, and mappings code to your own needs
  - Contract args/result/logs emissions tracking for successful receipts
  - Where to get help → TBD
- **Template Versions** → (EXAMPLES) not included in this grant, but referenced for possible future work.
  - Lite → FunctionCalls
  - Lite+ → FunctionCalls and Transfers
  - Granular → keep adding watchers for all the different types of calls
  - NFT API Standard → includes all of the schemas needed to track all the NEP-171 NFT collection events and sets up projects with an immediate NFT API for their project using the base template.
- **Walkthrough of Full Setup**
  - Installations
  - Setting up Hosted Service account
  - File configuration screenshots
    - package.json
    - subgraph.yaml
  - Deployment

- Testing your deployment with example queries
- *Everything will have screenshots*

Deliverables

Monorepo →

- subgraph.yaml file
- schema.subgraph file
- single mappings.ts file
- Documentation Readme.md file
- Milestone 1 tasks

Milestone 1

Tasks

Code

Below files and documents mentioned within this documents Technical Specifications.

- [ ] Clean and comment current code and push to open public “near-cast” repo
- [ ] subgraph.yaml file
- [ ] schema.graphql file
- [ ] mappings.ts file
- [ ] Readme.md file with Documentation

Documentation

Below files and documents mentioned within this documents Technical Specifications.

Sections

- [ ] NEAR CAST
- [ ] Prerequisites
- [ ] Quick Start
- [ ] Usage
- [ ] Query Reference
- [ ] Walkthrough of Full Setup

Query Reference

Get All OutcomeLogs of methodName

This query uses the methodName *ft_resolve_transfer* as an example query from the Meta Pool contract (Playground Link **here**)

Example Request

```
{
  searchFunction(text: "ft_resolve_transfer") {
    id
    methodName
    outcomeLogs
  }
}
```

Example Response


```
{
  "data": {
    "functionSearch": [
      {
        "id": "12mCHNqcvdWWSKQxTEMaoPk4furCrE7ii7tvKXcfiWGJ",
        "methodName": "ft_resolve_transfer",
        "outcomeLogs": "Refund 946080570296911801513110 from v1.jumbo_exchange.near to ras
as.near"
      },
      {
```

```
        "id": "...",
        "methodName": "...",
        "outcomeLogs": "..."
      {
    ]
  }
}
```

Grant Proposal Questions

This template was created as a prototyping tool for those curious about analyzing the more granular the on-chain activity of smart contracts deployed on NEAR. Find what calls are being emitted from the contract(s) you intend to index/create a subgraph for!

What are you building and why?

A configurable starting point for developers of all levels to analyze any on-chain activities of a NEAR smart contract before they start their next API build. This is purposed as a smart contract analytics tool using subgraphs on NEAR that is pre-configured to serve as a building block that enables rapid prototyping of any subgraphs on NEAR.

Why?

My colleague and I have been prototyping with this template throughout my projects without knowledge of how useful it was until realizing there was no longer a need to look at the smart contract code to build subgraphs on NEAR. This was amazing to me and broke down multiple barriers of jumping into more complex contract code to extract desired data. Although, I still recommend looking at contract code when developing indexing related tooling, but this is awesome as it is possible now to build indexing tools for on-chain activity by simply deploying a subgraph for it by just changing the account field in the YAML file! There were also no configurable starting templates or debug tools for NEAR subgraphs that my colleague or I were able to find.

What components of the project will be open source? Please provide as much detail as possible.

All of the components involved will be open source with anything included in the monorepo.

- subgraph.yaml file
- schema.subgraph file
- single mappings.ts file
- Documentation Readme.md file

Please provide an outline with the hours each component will take, as well as how long you expect this project to take?

subgraph.yaml and schema.graphql - 1 hour

mappings.ts - 1 week

- Refactoring current prototype code to be presentable, readable, commented - 1 day
- Adding mappings to return readable output for statusValue
- Adding mappings for events tracking with EventStandards
- Getting feedback on readability of layout

Documentation - 3 weeks to full completion

- Milestone 1 Documentation tasks

How will this impact the NEAR Ecosystem?

More developers making subgraphs on NEAR → more NEAR subgraphs being deployed, launchpad for new devs in ecosystem to index their projects data!! This makes developing subgraphs on NEAR so much easier and breaks down a barrier for

new devs thinking about building or indexing something on NEAR. The intention is that it makes other developers feel like you already have all the data and just have to format it. Because that is the case with this tool!

What will this grant help cover?

- Completion of Milestone 1 tasks
- Refining and critiquing the documentation to be ingested by incoming first time devs and subgraph devs
- Code maintenance for one month after completion
- I really want to dig into this so people can start using it and it's different versions regularly. The vision is that NEAR CAST will be a go-to solution to immediately bootstrap/prototype any future projects with relevant on-chain indexing on NEAR.

How will you measure success? Have you thought about any potential milestone?

After completion of Milestone 1 tasks I will do my best to market this on my twitter, share in discord channels, get the people going and understanding “how easy has become” to hook up indexing with NEAR’s Graph integration and how developers “don’t need to learn/be intimidated by Rust” to make subgraph’s on NEAR. A metric of success is the type of feedback received from developers in the space and if they are asking for examples, content, etc. if people ask for more, it could be interpreted that it is being well received and people are finding it useful.

Additional Comments

I have been working on API generation tools for NEAR and would love to chat about what I am building from this tooling. There is a lot of data included in The Graph NEAR integration and numerous ways to present it with another template, I would like to include that as it is needed and discuss if R&D type work would be relevant to future

work.
I am also interested in sharing future milestones to build this template/tooling out!

References & Contributors

Playground Link - NEAR Future Primal Contract (updated schema)

Playground Link - Meta Pool Contract

Playground Link - Tenk AstroDAO Contract

SimpleFi - Tarun - The debug log mappings that bundle receipts together.