EIP 1767: GraphQL interface to Ethereum node data (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 1: Data sheet for GraphQL interface to Ethereum node data

Title	GraphQL interface to Ethereum node data
Author	Nick Johnson, Raúl Kripalani, Kris Shinn
Status	Draft
Created	2019-02-14
Description	http://eips.ethereum.org/EIPS/eip-1767
Specification	http://eips.ethereum.org/EIPS/eip-1767#Specification
Category	Interface

Abstract

This EIP specifies a GraphQL schema for accessing data stored on an Ethereum node. It aims to provide a complete replacement to the read-only information exposed via the present JSON-RPC interface, while improving on usability, consistency, efficiency, and future-proofing.

Motivation

The current JSON-RPC interface for Ethereum nodes has a number of shortcomings. It's informally and incompletely specified in areas, which has led to incompatibilities around issues such as representation of empty byte strings ("" vs "0x" vs "0x0"), and it has to make educated guesses about the data a user will request, which often leads to unnecessary work.

For example, the `totalDifficulty` field is stored separately from the block header in common Ethereum node implementations, and many callers do not require this field. However, every call to `eth_getBlock` still retrieves this field, requiring a separate disk read, because the RPC server has no way of knowing if the user requires this field or not.

Similarly, transaction receipts in go-ethereum are stored on disk as a single binary blob for each block. Fetching a receipt for a single transaction requires fetching and deserializing this blob, then finding the relevant entry and returning it; this is accomplished by the `eth_getTransactionReceipt` API call. A common task for API consumers is to fetch all the receipts in a block; as a result, node implementations end up fetching and deserializing the same data repeatedly, leading to `O(n^2)` effort to fetch all transaction receipts from a block instead of `O(n)`.

Some of these issues could be fixed with changes to the existing JSON-RPC interface, at the cost of complicating the interface somewhat. Instead, we propose adopting a standard query language,