Kaiko

MIGRATION GUIDE

Contracts

Amberdata → Kaiko

2026

www.kaiko.com

1 At a glance

What this guide covers. The Amberdata contract details endpoint and the Kaiko endpoint that replaces it. Both return the verified name, the ABI and the verified source of a smart contract. Kaiko reads up to 50 contracts per request and adds the proxy resolution.

What is an ABI?

The ABI (Application Binary Interface) is the contract's public interface, published as a JSON array. Each entry describes one function, event, error or constructor: its name, its inputs and outputs with their types, and whether it changes state. A client needs the ABI to encode a call to the contract and to decode the transactions, return values and event logs it produces. The source is the verified Solidity or Vyper code the contract was compiled from.

What changes for you

- The contract address moves from the path to the query parameter `contract_addresses`, which takes 1 to 50 addresses in one call (§6).
- The blockchain is a required query parameter with a short code: `ethereum-mainnet` becomes `ethereum` (§4).
- The response wrapper changes to `data` plus `meta`; the single contract object becomes one row per contract (§3).
- `abi` and `source` carry the same content. `contract_name` is the verified contract name (`KittyCore`), no longer the token name (`CryptoKitties`) (§5).
- `is_proxy` and `implementation_address` are new (§5, §6).

Endpoint map

Amberdata endpoint	Kaiko endpoint	How to call it	§
`blockchains/contracts/{hash}`	`/blockchains/contracts`	`blockchain` + `contract_addresses`	6

Coverage & distribution

Data type: Verified smart contract details: name, ABI, source, proxy resolution

Blockchains: ethereum

Contracts: Contracts with source verified on Etherscan and collected by Kaiko

Distribution: REST API, `https://api.kaiko.io`

Availability: Live; response version `2026-10-01`

IDENTICAL no change RENAMED same idea, new name or format NEW Kaiko only REMOVED no equivalent

2 Endpoint & authentication

	Amberdata (today)	Kaiko (new)
Method	GET	GET

	Amberdata (today)	Kaiko (new)
Base URL	`https://api.amberdata.com`	`https://api.kaiko.io`
Path	`/blockchains/contracts/{hash}`	`/blockchains/contracts`
Auth	Header `x-api-key`	Header `X-Api-Key`, your Kaiko key. A missing header or a key in the query string returns 401 `missing_api_key`.
Blockchain	Header `x-amberdata-blockchain-id`, `ethereum-mainnet` only	Query `blockchain`, required, short code (§4)
Contracts per call	One, in the path	1 to 50, comma-separated in `contract_addresses`
Version	Not returned on this endpoint	Optional query `api_version=YYYY-MM-DD`; current version `2026-10-01`, echoed in `meta.api_version`
Parameters	None beyond the path and header	Send only the parameters listed in §6

3 Response envelope & metadata

The fields around the data. Amberdata returns one contract as an object under `payload`. Kaiko returns an array of contracts under `data`, one row per requested address.

Amberdata field	Kaiko field	Change	Notes
`status`	HTTP status line	REMOVED	Read the HTTP status code instead.
`title`		REMOVED	"OK" label.
`description`		REMOVED	Free-text status description.
`payload` (object)	`data` (array)	RENAMED	One row per contract, in ascending `contract_address` order. Read `data[0]` for a single contract.
	`meta.api_version`	NEW	Version the endpoint served.
	`meta.url`	NEW	The request as received, with `api_version` added.
	`meta.count`	NEW	Number of contracts in `data`.
	`meta.next_url`	NEW	Complete URL of the next page when the request names more contracts than `page_size`; `null` on the last page. Use it as returned.

Errors

Case	Amberdata	Kaiko
Address with no collected contract (unverified contract, wallet address, unused address)	500, empty message	404 `unknown_contract`

Case	Amberdata	Kaiko
Blockchain not served	400, "Blockchain Not Supported for this Endpoint"	404 `unsupported_blockchain` (`polygon` , `base` , `solana`)
Malformed or missing parameter, more than 50 addresses, `page_size` out of range		400 `invalid_parameter` naming the parameter
Missing API key		401 `missing_api_key`

- **Missing values.** Every key is always present; a value Kaiko does not have is `null` .
- **Error format.** Errors are `application/problem+json` with `status` , `title` , `detail` and a stable `code` . Branch on `code` , not on the text.

4 Blockchain & naming conventions

This guide's equivalent of exchange names: the identifiers you send and receive.

Blockchains

Amberdata	Kaiko code	Notes
`ethereum-mainnet`	`ethereum`	The one blockchain served on both APIs. Lower case: `Ethereum` and `ethereum-mainnet` return 400.

Conventions

- **Field names.** camelCase becomes snake_case: `contractName` → `contract_name` .
- **Addresses.** `0x` -prefixed, 40 hex characters, matched in any letter case, so an EIP-55 checksummed address works. Rows carry the address in lower case. An address sent twice in the list is returned once.
- **Paging.** `page_size` sets the contracts per page: default 10, maximum 50 (above it, 400). Follow `meta.next_url` until it is `null` . To read up to 50 contracts in one response, send `page_size=50` .
- **No time axis.** A contract is not a time series: the endpoint takes no time window and no sort order.
- **Response size.** Every row carries the full ABI and source. A single contract runs from under 10 KB to over 150 KB of source; size batches with that in mind.

5 Will your values change?

> **Short answer**
>
> The ABI is identical and the source text is the same. The contract name changes meaning.

- **ABI.** Same JSON array, entry for entry, served as JSON rather than as a string.
- **Contract name.** Amberdata `contractName` returns, for a token contract, the token's display name; Kaiko `contract_name` returns the name of the verified contract, as Etherscan publishes it: `CryptoKitties` → `KittyCore` , `USD//C` → `FiatTokenProxy` , `Wrapped Ether` → `WETH9` .
- **Proxies.** On both APIs a proxy's `abi` and `source` describe the proxy itself, not the implementation. Kaiko adds `is_proxy` and the `implementation_address` , which you can request in turn to read the implementation's ABI.

- **Contracts Amberdata could not serve.** Some verified contracts on which Amberdata returns 500 are served by Kaiko.
- **Contracts not collected.** An address Kaiko has not collected returns 404 `unknown_contract`: contracts with no verified source and addresses that hold no contract.

6 Contract details

Replaces. `blockchains/contracts/{hash}`.

What it gives you. For each requested contract: the verified name, the ABI as a JSON array, the verified Solidity or Vyper source, and whether the contract is a proxy with its implementation address.

	Amberdata (today)	Kaiko (new)
Path	`/blockchains/contracts/{hash}`	`/blockchains/contracts`

Request parameters

Amberdata param	Kaiko param	Change	Notes
`hash` (path)	`contract_addresses`	RENAMED	Required query parameter; 1 to 50 comma-separated addresses, any letter case. Send the list once: repeating the parameter returns 400.
`x-amberdata-blockchain-id` (header)	`blockchain`	RENAMED	Required query parameter, `ethereum` (§4).
	`page_size`	NEW	Contracts per page, 1 to 50, default 10.
	`api_version`	NEW	Optional, `YYYY-MM-DD`; a future date returns 400 `unsupported_api_version`.

Response data fields

Amberdata field	Kaiko field	Type	Change	Notes
	`blockchain`	string	NEW	Kaiko blockchain code, as requested.
`hash` (request path)	`contract_address`	string	NEW	On every row, lower case. Use it to match rows to your request.
`abi`	`abi`	array	IDENTICAL	Same entries (§5).
`contractName`	`contract_name`	string	RENAMED	Verified contract name, not the token name (§5).
`source`	`source`	string	IDENTICAL	Same text (§5).
`bytecode`		string	REMOVED	
	`is_proxy`	boolean	NEW	Whether Etherscan resolves the contract as a proxy.
	`implementation_address`	string	NEW	The proxy's implementation contract, lower case. Always present; `null` when the contract is not a proxy or the implementation was not resolved.

The schema allows `null` for `contract_name`, `abi` and `source`; in practice all three are set, since an address without verified source returns 404.

Before / after example

The CryptoKitties contract on Ethereum on both APIs. The ABI and the source are the same on both. The name differs: `CryptoKitties` on Amberdata, `KittyCore` on Kaiko. Both responses trimmed.

Amberdata

```
GET /blockchains/contracts/
    0x06012c8cf97bead5deae
    237070f9587f8e7a266d
x-api-key: <your-key>
x-amberdata-blockchain-id: ethereum-mainnet
```

Kaiko

```
GET https://api.kaiko.io
    /blockchains/contracts
    ?blockchain=ethereum
    &contract_addresses=
    0x06012c8cf97bead5deae
    237070f9587f8e7a266d
X-Api-Key: <your-key>
```

Amberdata


```
{
  "status": 200,
  "title": "OK",
  "description": "Successful request",
  "payload": {
    "bytecode":
      "0x606060409081526002...",
    "abi": [ {
      "constant": true,
      "inputs": [ {
        "name": "_interfaceID",
        "type": "bytes4"
      } ],
      "name": "supportsInterface",
      "outputs": [ {
        "name": "",
        "type": "bool"
      } ],
      "payable": false,
      "stateMutability": "view",
      "type": "function"
    }, ... ],
    "contractName": "CryptoKitties",
    "source": "pragma solidity ^0.4.11; ..."
  }
}
```


Kaiko


```
{
  "data": [ {
    "blockchain": "ethereum",
    "contract_address":
      "0x06012c8cf97bead5deae
       237070f9587f8e7a266d",
    "abi": [ {
      "constant": true,
      "inputs": [ {
        "name": "_interfaceID",
        "type": "bytes4"
      } ],
      "name": "supportsInterface",
      "outputs": [ {
        "name": "",
        "type": "bool"
      } ],
      "payable": false,
      "stateMutability": "view",
      "type": "function"
    }, ... ],
    "contract_name": "KittyCore",
    "source": "pragma solidity ^0.4.11; ...",
    "is_proxy": false,
    "implementation_address": null
  } ],
  "meta": { "api_version": "2026-10-01",
    "url": "...", "count": 1,
    "next_url": null }
}
```


Kaiko example: a proxy, read in a batch

USDC and USDT in one request with `page_size=1`, to show paging. Rows come in ascending address order, so USDC comes first; its `meta.next_url` serves USDT. USDC is a proxy: its ABI and source are those of the proxy, and `implementation_address` names the implementation. Response live, trimmed.

Kaiko request

```
GET https://api.kaiko.io
    /blockchains/contracts
    ?blockchain=ethereum
    &contract_addresses=
    0xA0b86991c6218b36c1D1
    9D4a2e9Eb0cE3606eB48,
    0xdac17f958d2ee523a220
    6206994597c13d831ec7
    &page_size=1
X-Api-Key: <your-key>
```

Kaiko response


```
{
  "data": [ {
    "blockchain": "ethereum",
    "contract_address":
      "0xa0b86991c6218b36c1d1
       9d4a2e9eb0ce3606eb48",
    "abi": [ {
      "constant": false,
      "inputs": [ {
        "name": "newImplementation",
        "type": "address"
      } ],
      "name": "upgradeTo",
      ...
    }, ... ],
    "contract_name": "FiatTokenProxy",
    "source": "pragma solidity ^0.4.24; ...",
    "is_proxy": true,
    "implementation_address":
      "0x43506849d7c04f9138d1
       a2050bbf3a0c054402dd"
  } ],
  "meta": { "api_version": "2026-10-01",
    "url": "...", "count": 1,
    "next_url": "https://api.kaiko.io
      /blockchains/contracts?..." }
}
```