Programmatic verification via the API

TheVera application uses a proxy-verifier service as an underlying backend. This service retrieves contract details for verification and sends them to the Ethereum bytecode database, which then writes them to the Verifier Alliance (Vera) database if verification succeeds.

The API provided by the proxy-verifier can be used to verify smart contracts. All endpoints are synchronous, meaning they wait for the operation to complete before returning any result.

All endpoints were originally designed for use by a frontend application. Currently there are no special endpoints for verification tools, but these may be developed in the future.

Workflow

There are 2 ways to verify Solidity and Vyper contracts using the service.

1. Providing a standard JSON input. This option is preferred, and it is recommended for use by automatic verification scripts.
2. Explicitly providing multiple files (multi-part).

Swagger documentation:https://blockscout.github.io/swaggers/services/proxy-verifier/index.html#/

Standard JSON

Solidity: POST https://blockscout.github.io/api/v1/solidity/sources:verify-standard-json

Vyper: POST https://blockscout.github.io/api/v1/vyper/sources:verify-standard-json

BothSolidity andVyper have the same standard JSON verification request and response structures.

Request

```

Copy { "contracts":[ { "chainId":"string", "address":"string" } ], "compiler":"string", "input":"string" }

```

Name Description contracts List of contracts to be verified. For each contract specify chainId and address. chainId Numeric ID of chain

List of supported chain IDs is available via:GET https://proxy-verifier.services.blockscout.com/api/v1/chainsMore info address 0x address of the contract to be verified compiler Compiler version. This can be retrieved via:

Solidity:GET https://proxy-verifier.services.blockscout.com/api/v1/solidity/compilers

Vyper:GET https://proxy-verifier.services.blockscout.com/api/v1/vyper/compilers Only one compiler version can be specified during verification.EVMVersions is only used formulti-part verification .More info input Standard JSON input encoded as a string - More info available here:https://docs.soliditylang.org/en/latest/using-the-compiler.html#input-description

Response

200 400 ```

Copy // Response (only a single response is returned, the other two will not be included) "contractValidationResults": { "items":[{ "message":"string", "status":"VALID/INVALID/INTERNAL_ERROR" }] }, "compilationFailure": { "message":"string" }, "contractVerificationResults": { "items":[{ "message":"string", "status":"PARTIALLY_VERIFIED/FULLY_VERIFIED/FAILURE/INTERNAL_ERROR" }] } }

Copy { "error":"BAD REQUEST" }

```

Success Response Outcomes

A successful response has 3 different outcomes; one is always returned:

- contractValidationResults
-
    - indicates that some (at least one) of the contracts do not support verification. This may mean that the address is not a contract, or that the contract at the given address self-destructed. For each provided contract, the
```

correspondingVALID / INVALID
- statuses with descriptions are returned in the same order the contracts were provided.
- compilationFailure
-
    - returned when the given source inputs cannot be compiled. This failure is ubiquitous and is not related to any particular provided contract. This means that the user provided invalid contract details during verification.
- contractVerificationResults
-
    - indicates that all provided contracts are valid and verification requests for each of them have been sent to theeth-bytecode-db
- service. The verification may result in partial or full matches, or verification failure if the contract cannot be verified with the provided source details. The results are returned in the same order as the provided contracts.
-

contractValidationResults compilationFailure contractVerificationResults ```

Copy { "contractValidationResults":{ "items":[{ "message":"Ok", "status":"VALID" }, { "message":"Address is not a smart-contract", "status":"INVALID" }, { "message":"Smart-contract was self-destructed", "status":"INVALID" }] } }

Copy { "compilationFailure":{ "message":"Compilation error: [\"No input sources specified.\"]" } }

// Another example { "compilationFailure":{ "message":"content is not a valid standard json: missing field`optimizer` at line 1 column 54" } }

Copy { "contractVerificationResults":{ "items":[{ "message":"https://eth-sepolia.blockscout.com/address/0x5ad88296ac87a138100e8793722a021f32e212d9", "status":"PARTIALLY_VERIFIED" }, { "message":"No contract could be verified with provided data", "status":"FAILURE" }] } }

```

Multi-part verification

Multi-part verification accepts source files explicitly along with some (the most common) configuration options. These options are limited, and in general, the standard JSON option is preferred.

Solidity

POST https://blockscout.github.io/api/v1/solidity/sources:verify-multi-part

```

Copy { "contracts":[{ "chainId":"string", "address":"string" }], "compiler":"string", "evmVersion":"string", "optimizationRuns":0, "sourceFiles":{ "filePath1":"string",// content of the file "filePath2":"string" }, "libraries":{ "libName1":"string",// 20 bytes hex encoded address "libName2":"string" } }

```

compiler andevmVersion must be valid and chosen from thelist of supported compilers and EVM versions . Otherwise, a 400 BAD REQUEST error will be returned.

TheoptimizationRuns option, if omitted, corresponds to the optimizer withenabled: false setup. If present, it should be a correct u32 value and result inruns: {optimizationRuns} setup withenabled: true .

sourceFiles is a mapping from file paths to corresponding contents.

libraries is a mapping from library names to their hex-encoded addresses.

Vyper

POST https://blockscout.github.io/api/v1/vyper/sources:verify-multi-part

```

Copy { "contracts":[{ "chainId":"string", "address":"string" }], "compiler":"string", "evmVersion":"string", "sourceFiles":{ "additionalProp1":"string", "additionalProp2":"string", "additionalProp3":"string" }, "interfaces":{ "additionalProp1":"string", "additionalProp2":"string", "additionalProp3":"string" } }

```

compiler andevmVersion must be valid and chosen from thelist of supported compilers and EVM versions . Otherwise, a 400 BAD REQUEST error will be returned.
```

sourceFiles is a mapping from file paths that should be specified inside thesources standard JSON option to corresponding contents.

interfaces is a mapping from file paths that should be specified inside theinterfaces standard JSON option to corresponding contents.

Additional Info

List of supported chains

GET http://proxy-verifier.services.blockscout.com/api/v1/chains

```

Copy // Response { "chains":[ { "id":"string", "name":"string", "iconUrl":"string" } ] }

```

Returns the list of supported chains with corresponding details. For verification purposes, only theid is important. These are the chain IDs supported by the service and allowed to be used during contract verification.

List of supported compiler versions

GET https://proxy-verifier.services.blockscout.com/api/v1/solidity/compilers

GET https://proxy-verifier.services.blockscout.com/api/v1/vyper/compilers

Returns the list of supported compiler and corresponding EVM versions. Only one of these compiler and EVM (if the multi-part option is used) versions are allowed to be specified during contract verification.

```

Copy { "compilers":[ { "version":"string", "evmVersions":[ "string" ] } ] }

```

Last updated5 months ago