
crytic-compile

Release 0.1.6

Mar 23, 2020

Contents:

1	crytic_compile	1
1.1	crytic_compile package	1
1.1.1	Subpackages	1
1.1.1.1	crytic_compile.compiler package	1
1.1.1.2	crytic_compile.cryticparser package	1
1.1.1.3	crytic_compile.platform package	2
1.1.1.4	crytic_compile.utils package	13
1.1.2	Submodules	17
1.1.3	crytic_compile.crytic_compile module	17
1.1.4	Module contents	21
2	Indices and tables	23
	Python Module Index	25
	Index	27

1.1 crytic_compile package

1.1.1 Subpackages

1.1.1.1 crytic_compile.compiler package

Submodules

crytic_compile.compiler.compiler module

Handle the compiler version

```
class crytic_compile.compiler.compiler.CompilerVersion(compiler, version, optimized)
```

Bases: tuple

compiler

Alias for field number 0

optimized

Alias for field number 2

version

Alias for field number 1

Module contents

1.1.1.2 crytic_compile.cryticparser package

Submodules

crytic_compile.cryticparser.cryticparser module

Module handling the cli arguments

`crytic_compile.cryticparser.cryticparser.init` (*parser: argparse.ArgumentParser*)

Add crytic-compile arguments to the parser

Parameters *parser* –

Returns

crytic_compile.cryticparser.defaults module

Default value for options

Module contents

Init module

1.1.1.3 crytic_compile.platform package

Submodules

crytic_compile.platform.abstract_platform module

Abstract Platform

`class crytic_compile.platform.abstract_platform.AbstractPlatform` (*target: str, **kwargs*)

Bases: object

This is the abstract class for the platform

HIDE = **False**

NAME = ''

PROJECT_URL = ''

TYPE = 0

`compile` (*crytic_compile: CryticCompile, **kwargs*)

Run the compilation

Parameters

- **crytic_compile** –
- **kwargs** –

Returns

`guessed_tests` () → List[str]

Guess the potential unit tests commands

Returns list of unit tests command guessed

`is_dependency` (*path: str*) → bool

Check if the target is a dependency

Parameters `path` –

Returns

static `is_supported(target: str, **kwargs) → bool`
 Check if the target is a project supported by this platform

Parameters `target` –

Returns

platform_name_used
 Return the underlying platform used

Returns

platform_project_url_used
 Return the underlying platform url used

Returns

platform_type_used
 Return the underlying platform url used

Returns

target
 Return the target name

Returns

exception `crytic_compile.platform.abstract_platform.IncorrectPlatformInitialization`
 Bases: `Exception`
 Exception raises if a platform was not properly defined

crytic_compile.platform.all_export module

Module containing all the supported export functions

crytic_compile.platform.all_platforms module

Module containing all the platforms

crytic_compile.platform.archive module

Archive platform. It is similar to the standard platform, except that the file generated contains a “source_content” field
 Which is a map: filename -> sourcecode

```
class crytic_compile.platform.archive.Archive(target: str, **kwargs)
    Bases: crytic_compile.platform.abstract_platform.AbstractPlatform
    Archive platform. It is similar to the Standard platform, but contains also the source code
    HIDE = True
    NAME = 'Archive'
    PROJECT_URL = 'https://github.com/crytic/crytic-compile'
    TYPE = 101
```

compile (*crytic_compile: CryticCompile, **kwargs*)
Compile

Parameters

- **crytic_compile** –
- **kwargs** –

Returns

is_dependency (*_path: str*) → bool
Check if the _path is a dependency. Always false

Parameters *_path* –

Returns

static is_supported (*target: str, **kwargs*) → bool
Check if the target is an archive

Parameters *target* –

Returns

crytic_compile.platform.archive.export_to_archive (*crytic_compile: CryticCompile, **kwargs*) → str

Export the archive

Parameters

- **crytic_compile** –
- **kwargs** –

Returns

crytic_compile.platform.archive.generate_archive_export (*crytic_compile: CryticCompile*) → *Tuple[Dict[KT, VT], str]*

Generate the archive export

Parameters *crytic_compile* –

Returns

crytic_compile.platform.brownie module

Brownie platform. <https://github.com/iamdefinitelyahuman/brownie>

class *crytic_compile.platform.brownie.Brownie* (*target: str, **kwargs*)
Bases: *crytic_compile.platform.abstract_platform.AbstractPlatform*

Brownie class

NAME = 'Brownie'

PROJECT_URL = 'https://github.com/iamdefinitelyahuman/brownie'

TYPE = 9

compile (*crytic_compile: CryticCompile, **kwargs*)
Compile the target

Parameters

- **crytic_compile** –
- **target** –
- **kwargs** –

Returns

is_dependency (*_path: str*) → bool
Check if the path is a dependency

Parameters *_path* –

Returns

static is_supported (*target: str, **kwargs*) → bool
Check if the target is a brownie env

Parameters *target* –

Returns**crytic_compile.platform.dapp module**

Dapp platform. <https://github.com/dapphub/dapptools>

class `crytic_compile.platform.dapp.Dapp` (*target: str, **kwargs*)
Bases: `crytic_compile.platform.abstract_platform.AbstractPlatform`

Dapp class

NAME = 'Dapp'

PROJECT_URL = 'https://github.com/dapphub/dapptools'

TYPE = 4

compile (*crytic_compile: CryticCompile, **kwargs*)
Compile the target

Parameters

- **crytic_compile** –
- **target** –
- **kwargs** –

Returns

is_dependency (*path: str*) → bool
Check if the path is a dependency

Parameters *path* –

Returns

static is_supported (*target: str, **kwargs*) → bool
Heuristic used: check if “dapp build” is present in Makefile

Parameters *target* –

Returns

crytic_compile.platform.embark module

Embark platform. <https://github.com/embark-framework/embark>

```
class crytic_compile.platform.embark.Embark (target: str, **kwargs)
    Bases: crytic_compile.platform.abstract_platform.AbstractPlatform
    Embark platform
    NAME = 'Embark'
    PROJECT_URL = 'https://github.com/embarklabs/embark'
    TYPE = 3
    compile (crytic_compile: CryticCompile, **kwargs)
        Compile the target

        Parameters
            • crytic_compile –
            • target –
            • kwargs –

        Returns

    is_dependency (path: str) → bool
        Check if the path is a dependency

        Parameters path –

        Returns

    static is_supported (target: str, **kwargs) → bool
        Check if the target is an embark project

        Parameters target –

        Returns
```

crytic_compile.platform.etherlime module

Etherlime platform. <https://github.com/LimeChain/etherlime>

```
class crytic_compile.platform.etherlime.Etherlime (target: str, **kwargs)
    Bases: crytic_compile.platform.abstract_platform.AbstractPlatform
    Etherlime platform
    NAME = 'Etherlime'
    PROJECT_URL = 'https://github.com/LimeChain/etherlime'
    TYPE = 5
    compile (crytic_compile: CryticCompile, **kwargs)
        Compile the target

        Parameters
            • crytic_compile –
            • target –
```

- **kwargs** –

Returns

is_dependency (*path: str*) → bool

Check if the path is a dependency

Parameters path –

Returns

static is_supported (*target: str, **kwargs*) → bool

Check if the target is an etherlime project

Parameters target –

Returns

crytic_compile.platform.etherscan module

Etherscan platform.

class crytic_compile.platform.etherscan.**Etherscan** (*target: str, **kwargs*)
 Bases: *crytic_compile.platform.abstract_platform.AbstractPlatform*

Etherscan platform

NAME = 'Etherscan'

PROJECT_URL = 'https://etherscan.io/'

TYPE = 6

compile (*crytic_compile: CryticCompile, **kwargs*)

Compile the tharget :param crytic_compile: :param target: :param kwargs: :return:

is_dependency (*_path: str*) → bool

Always return false

Parameters _path –

Returns

static is_supported (*target: str, **kwargs*) → bool

Check if the target is an etherscan address

Parameters target –

Returns

crytic_compile.platform.etherscan.convert_version (*version: str*) → str

Convert the compiler version :param version: :return:

crytic_compile.platform.exceptions module

Crytic Compile Exceptions

exception crytic_compile.platform.exceptions.**InvalidCompilation**

Bases: *Exception*

Invalid compilation exception

crytic_compile.platform.solc module

Solc platform

```
class crytic_compile.platform.solc.Solc (target: str, **kwargs)
    Bases: crytic_compile.platform.abstract_platform.AbstractPlatform
    Solc platform
    NAME = 'solc'
    PROJECT_URL = 'https://github.com/ethereum/solidity'
    TYPE = 1
    compile (crytic_compile: CryticCompile, **kwargs)
        Compile the target
        Parameters
            • crytic_compile –
            • kwargs –
        Returns
    is_dependency (_path: str) → bool
        Always return false
        Parameters _path –
        Returns
    static is_supported (target: str, **kwargs) → bool
        Check if the target is a solc project
        Parameters target –
        Returns
    crytic_compile.platform.solc.export_to_solc (crytic_compile: CryticCompile, **kwargs)
        Export the project to the solc format
        → Optional[str]
        Parameters
            • crytic_compile –
            • kwargs –
        Returns
    crytic_compile.platform.solc.get_version (solc: str) → str
        Get the compiler version used
        Parameters solc –
        Returns
    crytic_compile.platform.solc.is_optimized (solc_arguments: str) → bool
        Check if optimization are used
        Parameters solc_arguments –
        Returns
    crytic_compile.platform.solc.relative_to_short (relative)
        Convert relative to short
```

Parameters *relative* –

Returns

`crytic_compile.platform.solc_standard_json` module

Handle compilation through the standard solc json format

```
class crytic_compile.platform.solc_standard_json.SolcStandardJson (target:
                                                    Union[str,
                                                    dict]      =
                                                    None,
                                                    **kwargs)
```

Bases: `crytic_compile.platform.solc.Solc`

Represent the Standard solc Json object

NAME = 'Solc-json'

PROJECT_URL = 'https://solidity.readthedocs.io/en/latest/using-the-compiler.html#compiling'

TYPE = 10

add_remapping (*remapping: str*)

Append our remappings

Parameters *remapping* –

Returns

add_source_file (*file_path: str*)

Append file

Parameters *file_path* –

Returns

compile (*crytic_compile: CryticCompile, **kwargs*)

Compile the target

Parameters

- *crytic_compile* –
- *target* –
- *kwargs* –

Returns

to_dict () → Dict[KT, VT]

Patch in our desired output types

Returns

`crytic_compile.platform.standard` module

Standard crytic-compile export

```
class crytic_compile.platform.standard.Standard (target: str, **kwargs)
    Bases: crytic_compile.platform.abstract_platform.AbstractPlatform
```

Standard platform (crytic-compile specific)

HIDE = True

NAME = 'Standard'

PROJECT_URL = 'https://github.com/crytic/crytic-compile'

TYPE = 100

compile (*crytic_compile: CryticCompile, **kwargs*)

Compile the target (load file)

Parameters

- **crytic_compile** –
- **target** –
- **kwargs** –

Returns

is_dependency (*path: str*) → bool

Always return False

Parameters **path** –

Returns

static is_supported (*target: str, **kwargs*) → bool

Check if the target is the standard crytic compile export

Parameters **target** –

Returns

platform_name_used

Return the underlying platform used

Returns

platform_project_url_used

Return the underlying platform url used

Returns

platform_type_used

Return the underlying platform url used

Returns

crytic_compile.platform.standard.export_to_standard (*crytic_compile: CryticCompile, **kwargs*) → str

Export the project to the standard crytic compile format :param crytic_compile: :param kwargs: :return:

crytic_compile.platform.standard.generate_standard_export (*crytic_compile: CryticCompile*) → Dict[KT, VT]

Export the standard crytic compile export

Parameters **crytic_compile** –

Returns

crytic_compile.platform.standard.load_from_compile (*crytic_compile: CryticCompile, loaded_json: Dict[KT, VT]*) → Tuple[int, List[str]]

Load from json

Parameters

- **crytic_compile** –
- **loaded_json** –

Returns**crytic_compile.platform.truffle module**

Truffle platform

class `crytic_compile.platform.truffle.Truffle` (*target: str, **kwargs*)
 Bases: `crytic_compile.platform.abstract_platform.AbstractPlatform`

Truffle platform

NAME = 'Truffle'

PROJECT_URL = 'https://github.com/trufflesuite/truffle'

TYPE = 2

compile (*crytic_compile: CryticCompile, **kwargs*)
 Compile the target

Parameters **kwargs** –

Returns

is_dependency (*path: str*) → bool
 Check if the target is a dependency

Parameters **path** –

Returns

static is_supported (*target: str, **kwargs*) → bool
 Check if the target is a truffle project

Parameters **target** –

Returns

`crytic_compile.platform.truffle.export_to_truffle` (*crytic_compile: CryticCompile, **kwargs*) → Optional[str]

Export to the truffle format

Parameters

- **crytic_compile** –
- **kwargs** –

Returns**crytic_compile.platform.types module**

Handle the platform type

class `crytic_compile.platform.types.Type`
 Bases: `enum.IntEnum`

Represent the different platform

```

ARCHIVE = 101
BROWNIE = 9
DAPP = 4
EMBARK = 3
ETHERLIME = 5
ETHERSCAN = 6
NOT_IMPLEMENTED = 0
SOLC = 1
SOLC_STANDARD_JSON = 10
STANDARD = 100
TRUFFLE = 2
VYPER = 7
WAFFLE = 8

```

crytic_compile.platform.vyper module

Vyper platform

```

class crytic_compile.platform.vyper.Vyper(target: str, **kwargs)
    Bases: crytic_compile.platform.abstract_platform.AbstractPlatform

```

Vyper platform

```
NAME = 'vyper'
```

```
PROJECT_URL = 'https://github.com/vyperlang/vyper'
```

```
TYPE = 7
```

```
compile(crytic_compile: CryticCompile, **kwargs)
    Compile the target
```

Parameters

- **crytic_compile** –
- **target** –
- **kwargs** –

Returns

```
is_dependency(_path)
    Always return false
```

Parameters **_path** –

Returns

```
static is_supported(target: str, **kwargs) → bool
    Check if the target is a vyper project
```

Parameters **target** –

Returns

crytic_compile.platform.waffle module

Waffle platform

```
class crytic_compile.platform.waffle.Waffle (target: str, **kwargs)
    Bases: crytic_compile.platform.abstract_platform.AbstractPlatform
```

Waffle platform

NAME = 'Waffle'

PROJECT_URL = 'https://github.com/EthWorks/Waffle'

TYPE = 8

compile (crytic_compile: *CryticCompile*, **kwargs)
Compile the target

Parameters

- **crytic_compile** –
- **target** –
- **kwargs** –

Returns

is_dependency (path: str) → bool
Check if the path is a dependency

Parameters path –

Returns

static is_supported (target: str, **kwargs) → bool
Check if the target is a waffle project

Parameters target –

Returns

Module contents

Init module

1.1.1.4 crytic_compile.utils package

Submodules

crytic_compile.utils.naming module

Module handling the file naming operation (relative -> absolute, etc)

```
class crytic_compile.utils.naming.Filename (absolute, used, relative, short)
    Bases: tuple
```

absolute

Alias for field number 0

relative

Alias for field number 2

short

Alias for field number 3

used

Alias for field number 1

`cryptic_compile.utils.naming.combine_filename_name(filename: str, name: str)`

Combine the filename with the contract name

Parameters

- **filename** –
- **name** –

Returns

`cryptic_compile.utils.naming.convert_filename(used_filename: Union[str, pathlib.Path], relative_to_short, cryptic_compile: CrypticCompile, working_dir=None) → cryptic_compile.utils.naming.Filename`

Convert filename. The used_filename can be absolute, relative, or missing node_modules/contracts directory
convert_filename return a tuple(absolute,used), where absolute points to the absolute path, and used the original

Parameters

- **used_filename** –
- **relative_to_short** – lambda function
- **cryptic_compile** –
- **working_dir** –

Returns Filename (namedtuple)

`cryptic_compile.utils.naming.extract_filename(name: str)`

Convert '/path:Contract' to /path

`cryptic_compile.utils.naming.extract_name(name: str)`

Convert '/path:Contract' to Contract

cryptic_compile.utils.natspec module

Natspec module <https://solidity.readthedocs.io/en/latest/natspec-format.html>

class `cryptic_compile.utils.natspec.DevDoc(devdoc: Dict[KT, VT])`

Bases: object

Model the dev doc

author

Return the dev author

Returns Optional[str]

details

Return the dev details

Returns Optional[str]

```

export () → Dict[KT, VT]
    Export to a python dict

    Returns Dict

methods
    Return the dev methods

    Returns Dict[str, DevMethod]

title
    Return the dev title

    Returns Optional[str]

class crytic_compile.utils.natspec.DevMethod(method)
    Bases: object

    Model the dev method

    author
        Return the method author

        Returns Optional[str]

    details
        Return the method details

        Returns Optional[str]

    export () → Dict[KT, VT]
        Export to a python dict

        Returns Dict

    method_return
        Return the method return

        Returns Optional[str]

    params
        Return the method params

        Returns Dict[str, str]

class crytic_compile.utils.natspec.Natspec(userdoc: Dict[KT, VT], devdoc: Dict[KT, VT])
    Bases: object

    Model natspec

    devdoc
        Return the devdoc

        Returns DevDoc

    userdoc
        Return the userdoc

        Returns UserDoc

class crytic_compile.utils.natspec.UserDoc(userdoc: dict)
    Bases: object

    Model the user doc

```

export () → Dict[KT, VT]
Export to a python dict

Returns Dict

methods

Return the user methods

Returns Dict[str, UserMethod]

notice

Return the user notice

Returns Optional[str]

class cryptic_compile.utils.natspec.**UserMethod**(*method*)

Bases: object

Model the user method

export () → Dict[KT, VT]
Export to a python dict

Returns Dict

notice

Return the method notice

Returns Optional[str]

cryptic_compile.utils.npm module

Module handling NPM related features

cryptic_compile.utils.npm.**get_package_name**(*target_txt: Union[str, SolcStandardJson]*) → Optional[str]

Return the package's name

Parameters *target_txt* –

Returns str or None

cryptic_compile.utils.unit_tests module

Module handling unit-tests features

cryptic_compile.utils.unit_tests.**guess_tests**(*target: str*) → List[str]

Try to guess the unit tests

Parameters *target* –

Returns

cryptic_compile.utils.zip module

Handle ZIP operations

cryptic_compile.utils.zip.**load_from_zip**(*target: str*) → List[cryptic_compile.cryptic_compile.CrypticCompile]

Load a file from a zip

Parameters *target* –

Returns

`crytic_compile.utils.zip.save_to_zip` (*crytic_compiles: List[CryticCompile], zipfile: str*)
 Save projects to a zip

Parameters

- **crytic_compiles** –
- **zipfile** –

Returns**Module contents****1.1.2 Submodules****1.1.3 crytic_compile.crytic_compile module**

CryticCompile main module. Handle the compilation.

class `crytic_compile.crytic_compile.CryticCompile` (*target: Union[str, crytic_compile.platform.abstract_platform.AbstractPlatform], **kwargs*)

Bases: `object`

Main class.

abi (*name: str*) → `Dict[KT, VT]`
 Get the ABI from a contract

Parameters **name** –

Returns

abis
 Return the ABIs

Returns

absolute_filename_of_contract (*name: str*) → `str`

Returns Absolute filename

ast (*path: str*) → `Optional[Dict[KT, VT]]`
 Return of the file

Parameters **path** –

Returns

asts
 Return the ASTs

Returns `dict` (absolute filename -> AST)

bytecode_init (*name: str, libraries: Union[None, Dict[str, str]] = None*) → `str`
 Return the init bytecode of the contract. If library is provided, patch the bytecode

Parameters

- **name** –
- **libraries** –

Returns

bytecode_only

Return true if only the bytecode was retrieved

Returns

bytecode_runtime (*name: str, libraries: Union[None, Dict[str, str]] = None*) → str

Return the runtime bytecode of the contract. If library is provided, patch the bytecode

Parameters

- **name** –
- **libraries** –

Returns

bytecodes_init

Return the init bytecodes

Returns

bytecodes_runtime

Return the runtime bytecodes

Returns

compiler_version

Return the compiler used as a namedtuple(compiler, version)

Returns

contracts_absolute_filenames

Return a dict (contract_name -> absolute filename)

Returns

contracts_filenames

Return a dict contract_name -> Filename namedtuple (absolute, used)

Returns dict(name -> utils.namings.Filename)

contracts_names

Return the contracts names

Returns

contracts_names_without_libraries

Return the contracts names (without the libraries)

Returns

dependencies

Return the dependencies files

Returns

export (***kwargs*) → Optional[str]

Export to json. The json format can be crytic-compile, solc or truffle.

filename_lookup (*filename: str*) → crytic_compile.utils.naming.Filename

Return a crytic_compile.naming.Filename from a any filename form (used/absolute/relative)

Parameters **filename** – str

Returns crytic_compile.naming.Filename

filename_of_contract (*name: str*) → cryptic_compile.utils.naming.Filename

Returns utils.namings.Filename

filenames

Returns set(naming.Filename)

find_absolute_filename_from_used_filename (*used_filename: str*) → str

Return the absolute filename based on the used one

Parameters used_filename –

Returns absolute filename

hashes (*name: str*) → Dict[str, int]

Return the hashes of the functions

Parameters name –

Returns

static import_archive_compilations (*compiled_archive: Union[str, Dict[KT, VT]]*) → List[cryptic_compile.cryptic_compile.CrypticCompile]

Import from an archive. compiled_archive is either a json file or the loaded dictionary The dictionary must contain the “compilations” keyword

Parameters compiled_archive –

Returns

is_dependency (*filename: str*) → bool

Check if the filename is a dependency

Parameters filename –

Returns

libraries

Return the libraries used (contract_name -> [(library, pattern)])

Returns

libraries_names (*name: str*) → List[str]

Return the name of the libraries used by the contract

Parameters name – contract

Returns list of libraries name

libraries_names_and_patterns (*name*)

Return the name of the libraries used by the contract

Parameters name – contract

Returns list of (libraries name, pattern)

natspec

Return the natspec of the contractse

Returns Dict[str, Natspec]

package

Return the package name

Returns

package_name

Returns str or None

platform

Return the platform module

Returns

relative_filename_from_absolute_filename (*absolute_filename: str*) → str

Return the relative file based on the absolute name

Parameters **absolute_filename** –

Returns

src_content

Return the source content, filename -> source_code

Returns

src_content_for_file (*filename_absolute: str*) → Optional[str]

Get the source code of the file

Parameters **filename_absolute** –

Returns

srcmap_init (*name: str*) → List[str]

Return the init srcmap

Parameters **name** –

Returns

srcmap_runtime (*name: str*) → List[str]

Return the runtime srcmap

Parameters **name** –

Returns

srcmaps_init

Return the init srcmap

Returns

srcmaps_runtime

Return the runtime srcmap

Returns

target

Return the target (project)

Returns

type

Return the type of the platform used

Returns

used_filename_of_contract (*name: str*) → str

Returns Used filename

working_dir

Return the working dir

Returns

`crytic_compile.crytic_compile.compile_all` (*target*: *str*, ***kwargs*) → `List[crytic_compile.crytic_compile.CryticCompile]`

Given a direct or glob pattern target, compiles all underlying sources and returns all the relevant instances of CryticCompile.

Parameters

- **target** – A string representing a file/directory path or glob pattern denoting where compilation should occur.
- **kwargs** – The remainder of the arguments passed through to all compilation steps.

Returns Returns a list of CryticCompile instances for all compilations which occurred.

`crytic_compile.crytic_compile.get_platforms` () → `List[Type[crytic_compile.platform.abstract_platform.AbstractPlatform]]`
Return the available platforms classes

Returns

`crytic_compile.crytic_compile.is_supported` (*target*: *str*) → `bool`
Check if the target is supported

Parameters **target** –

Returns

1.1.4 Module contents

Init module

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

C

- [crytic_compile](#), 21
- [crytic_compile.compiler](#), 1
- [crytic_compile.compiler.compiler](#), 1
- [crytic_compile.crytic_compile](#), 17
- [crytic_compile.cryticparser](#), 2
- [crytic_compile.cryticparser.cryticparser](#), 2
- [crytic_compile.cryticparser.defaults](#), 2
- [crytic_compile.platform](#), 13
- [crytic_compile.platform.abstract_platform](#), 2
- [crytic_compile.platform.all_export](#), 3
- [crytic_compile.platform.all_platforms](#), 3
- [crytic_compile.platform.archive](#), 3
- [crytic_compile.platform.brownie](#), 4
- [crytic_compile.platform.dapp](#), 5
- [crytic_compile.platform.embark](#), 6
- [crytic_compile.platform.etherlime](#), 6
- [crytic_compile.platform.etherscan](#), 7
- [crytic_compile.platform.exceptions](#), 7
- [crytic_compile.platform.solc](#), 8
- [crytic_compile.platform.solc_standard_json](#), 9
- [crytic_compile.platform.standard](#), 9
- [crytic_compile.platform.truffle](#), 11
- [crytic_compile.platform.types](#), 11
- [crytic_compile.platform.vyper](#), 12
- [crytic_compile.platform.waffle](#), 13
- [crytic_compile.utils](#), 17
- [crytic_compile.utils.naming](#), 13
- [crytic_compile.utils.natspec](#), 14
- [crytic_compile.utils.npm](#), 16
- [crytic_compile.utils.unit_tests](#), 16
- [crytic_compile.utils.zip](#), 16

A

`abi()` (*crytic_compile.crytic_compile.CryticCompile* method), 17
`abis` (*crytic_compile.crytic_compile.CryticCompile* attribute), 17
`absolute` (*crytic_compile.utils.naming.Filename* attribute), 13
`absolute_filename_of_contract()` (*crytic_compile.crytic_compile.CryticCompile* method), 17
`AbstractPlatform` (class in *crytic_compile.platform.abstract_platform*), 2
`add_remapping()` (*crytic_compile.platform.solc_standard_json.SolcStandardJson* method), 9
`add_source_file()` (*crytic_compile.platform.solc_standard_json.SolcStandardJson* method), 9
`Archive` (class in *crytic_compile.platform.archive*), 3
`ARCHIVE` (*crytic_compile.platform.types.Type* attribute), 11
`ast()` (*crytic_compile.crytic_compile.CryticCompile* method), 17
`asts` (*crytic_compile.crytic_compile.CryticCompile* attribute), 17
`author` (*crytic_compile.utils.natspec.DevDoc* attribute), 14
`author` (*crytic_compile.utils.natspec.DevMethod* attribute), 15

B

`Brownie` (class in *crytic_compile.platform.brownie*), 4
`BROWNIE` (*crytic_compile.platform.types.Type* attribute), 12
`bytecode_init()` (*crytic_compile.crytic_compile.CryticCompile* method), 17
`bytecode_only` (*crytic_compile.crytic_compile.CryticCompile* attribute), 18

`bytecode_runtime()` (*crytic_compile.crytic_compile.CryticCompile* method), 18
`bytecodes_init` (*crytic_compile.crytic_compile.CryticCompile* attribute), 18
`bytecodes_runtime` (*crytic_compile.crytic_compile.CryticCompile* attribute), 18

C

`combine_filename_name()` (in module *crytic_compile.utils.naming*), 14
`compile()` (*crytic_compile.platform.abstract_platform.AbstractPlatform* method), 2
`compile()` (*crytic_compile.platform.archive.Archive* method), 3
`compile()` (*crytic_compile.platform.brownie.Brownie* method), 4
`compile()` (*crytic_compile.platform.dapp.Dapp* method), 5
`compile()` (*crytic_compile.platform.embark.Embark* method), 6
`compile()` (*crytic_compile.platform.etherlime.Etherlime* method), 6
`compile()` (*crytic_compile.platform.etherscan.Etherscan* method), 7
`compile()` (*crytic_compile.platform.solc.Solc* method), 8
`compile()` (*crytic_compile.platform.solc_standard_json.SolcStandardJson* method), 9
`compile()` (*crytic_compile.platform.standard.Standard* method), 10
`compile()` (*crytic_compile.platform.truffle.Truffle* method), 11
`compile()` (*crytic_compile.platform.vyper.Vyper* method), 12

`compile()` (`cryptic_compile.platform.waffle.Waffle` `method`), 13
`compile_all()` (`in module cryptic_compile.cryptic_compile`), 20
`compiler(cryptic_compile.compiler.compiler.CompilerVersion` `(module)`, 9 `attribute`), 1
`compiler_version` (`cryptic_compile.cryptic_compile.CrypticCompile` `attribute`), 18
`CompilerVersion` (`class in cryptic_compile.compiler.compiler`), 1
`contracts_absolute_filenames` (`cryptic_compile.cryptic_compile.CrypticCompile` `attribute`), 18
`contracts_filenames` (`cryptic_compile.cryptic_compile.CrypticCompile` `attribute`), 18
`contracts_names` (`cryptic_compile.cryptic_compile.CrypticCompile` `attribute`), 18
`contracts_names_without_libraries` (`cryptic_compile.cryptic_compile.CrypticCompile` `attribute`), 18
`convert_filename()` (`in module cryptic_compile.utils.naming`), 14
`convert_version()` (`in module cryptic_compile.platform.etherscan`), 7
`cryptic_compile` (`module`), 21
`cryptic_compile.compiler` (`module`), 1
`cryptic_compile.compiler.compiler` (`module`), 1
`cryptic_compile.cryptic_compile` (`module`), 17
`cryptic_compile.crypticparser` (`module`), 2
`cryptic_compile.crypticparser.crypticparser` (`module`), 2
`cryptic_compile.crypticparser.defaults` (`module`), 2
`cryptic_compile.platform` (`module`), 13
`cryptic_compile.platform.abstract_platform` (`module`), 2
`cryptic_compile.platform.all_export` (`module`), 3
`cryptic_compile.platform.all_platforms` (`module`), 3
`cryptic_compile.platform.archive` (`module`), 3
`cryptic_compile.platform.brownie` (`module`), 4
`cryptic_compile.platform.dapp` (`module`), 5
`cryptic_compile.platform.embark` (`module`), 6
`cryptic_compile.platform.etherlime` (`module`), 6
`cryptic_compile.platform.etherscan` (`module`), 7
`cryptic_compile.platform.exceptions` (`module`), 7
`cryptic_compile.platform.solc` (`module`), 8
`cryptic_compile.platform.solc_standard_json` (`module`), 9
`cryptic_compile.platform.standard` (`module`), 9
`cryptic_compile.platform.truffle` (`module`), 11
`cryptic_compile.platform.types` (`module`), 11
`cryptic_compile.platform.vyper` (`module`), 12
`cryptic_compile.platform.waffle` (`module`), 13
`cryptic_compile.utils` (`module`), 17
`cryptic_compile.utils.naming` (`module`), 13
`cryptic_compile.utils.natspec` (`module`), 14
`cryptic_compile.utils.npm` (`module`), 16
`cryptic_compile.utils.unit_tests` (`module`), 16
`cryptic_compile.utils.zip` (`module`), 16
`CrypticCompile` (`class in cryptic_compile.cryptic_compile`), 17

D

`Dapp` (`class in cryptic_compile.platform.dapp`), 5
`DAPP` (`cryptic_compile.platform.types.Type` `attribute`), 12
`dependencies` (`cryptic_compile.cryptic_compile.CrypticCompile` `attribute`), 18
`details` (`cryptic_compile.utils.natspec.DevDoc` `attribute`), 14
`details` (`cryptic_compile.utils.natspec.DevMethod` `attribute`), 15
`DevDoc` (`class in cryptic_compile.utils.natspec`), 14
`devdoc` (`cryptic_compile.utils.natspec.Natspec` `attribute`), 15
`DevMethod` (`class in cryptic_compile.utils.natspec`), 15

E

`Embark` (`class in cryptic_compile.platform.embark`), 6
`EMBARK` (`cryptic_compile.platform.types.Type` `attribute`), 12
`Etherlime` (`class in cryptic_compile.platform.etherlime`), 6
`ETHERLIME` (`cryptic_compile.platform.types.Type` `attribute`), 12
`Etherscan` (`class in cryptic_compile.platform.etherscan`), 7
`ETHERSCAN` (`cryptic_compile.platform.types.Type` `attribute`), 12
`export()` (`cryptic_compile.cryptic_compile.CrypticCompile` `method`), 18
`export()` (`cryptic_compile.utils.natspec.DevDoc` `method`), 14

export () (crytic_compile.utils.natspec.DevMethod method), 15

export () (crytic_compile.utils.natspec.UserDoc method), 15

export () (crytic_compile.utils.natspec.UserMethod method), 16

export_to_archive () (in module crytic_compile.platform.archive), 4

export_to_solc () (in module crytic_compile.platform.solc), 8

export_to_standard () (in module crytic_compile.platform.standard), 10

export_to_truffle () (in module crytic_compile.platform.truffle), 11

extract_filename () (in module crytic_compile.utils.naming), 14

extract_name () (in module crytic_compile.utils.naming), 14

F

Filename (class in crytic_compile.utils.naming), 13

filename_lookup () (crytic_compile.crytic_compile.CryticCompile method), 18

filename_of_contract () (crytic_compile.crytic_compile.CryticCompile method), 18

filenames (crytic_compile.crytic_compile.CryticCompile attribute), 19

find_absolute_filename_from_used_filename () (crytic_compile.crytic_compile.CryticCompile method), 19

G

generate_archive_export () (in module crytic_compile.platform.archive), 4

generate_standard_export () (in module crytic_compile.platform.standard), 10

get_package_name () (in module crytic_compile.utils.npm), 16

get_platforms () (in module crytic_compile.crytic_compile), 21

get_version () (in module crytic_compile.platform.solc), 8

guess_tests () (in module crytic_compile.utils.unit_tests), 16

guessed_tests () (crytic_compile.platform.abstract_platform.AbstractPlatform method), 2

H

hashes () (crytic_compile.crytic_compile.CryticCompile method), 19

is_optimized() (in module cry-
tic_compile.platform.solc), 8

is_supported() (cry-
tic_compile.platform.abstract_platform.AbstractPlatform
static method), 3

is_supported() (cry-
tic_compile.platform.archive.Archive
method), 4

is_supported() (cry-
tic_compile.platform.brownie.Brownie
method), 5

is_supported() (cry-
tic_compile.platform.dapp.Dapp
method), 5

is_supported() (cry-
tic_compile.platform.embark.Embark
method), 6

is_supported() (cry-
tic_compile.platform.etherlime.Etherlime
static method), 7

is_supported() (cry-
tic_compile.platform.etherscan.Etherscan
static method), 7

is_supported() (cryptic_compile.platform.solc.Solc
static method), 8

is_supported() (cry-
tic_compile.platform.standard.Standard
method), 10

is_supported() (cry-
tic_compile.platform.truffle.Truffle
method), 11

is_supported() (cry-
tic_compile.platform.vyper.Vyper
method), 12

is_supported() (cry-
tic_compile.platform.waffle.Waffle
method), 13

is_supported() (in module cry-
tic_compile.cryptic_compile), 21

L

libraries (cryptic_compile.cryptic_compile.CrypticCompile
attribute), 19

libraries_names() (cry-
tic_compile.cryptic_compile.CrypticCompile
method), 19

libraries_names_and_patterns() (cry-
tic_compile.cryptic_compile.CrypticCompile
method), 19

load_from_compile() (in module cry-
tic_compile.platform.standard), 10

load_from_zip() (in module cry-
tic_compile.utils.zip), 16

M

method_return (cry-
tic_compile.utils.natspec.DevMethod at-
tribute), 15

methods (cryptic_compile.utils.natspec.DevDoc at-
tribute), 15

methods (cryptic_compile.utils.natspec.UserDoc at-
tribute), 16

N

NAME (cryptic_compile.platform.abstract_platform.AbstractPlatform
attribute), 2

NAME (cryptic_compile.platform.archive.Archive at-
tribute), 3

NAME (cryptic_compile.platform.brownie.Brownie at-
tribute), 4

NAME (cryptic_compile.platform.dapp.Dapp attribute), 5

NAME (cryptic_compile.platform.embark.Embark at-
tribute), 6

NAME (cryptic_compile.platform.etherlime.Etherlime at-
tribute), 6

NAME (cryptic_compile.platform.etherscan.Etherscan at-
tribute), 7

NAME (cryptic_compile.platform.solc.Solc attribute), 8

NAME (cryptic_compile.platform.solc_standard_json.SolcStandardJson
attribute), 9

NAME (cryptic_compile.platform.standard.Standard
attribute), 10

NAME (cryptic_compile.platform.truffle.Truffle attribute),
11

NAME (cryptic_compile.platform.vyper.Vyper attribute), 12

NAME (cryptic_compile.platform.waffle.Waffle attribute),
13

Natspec (class in cryptic_compile.utils.natspec), 15

natspec (cryptic_compile.cryptic_compile.CrypticCompile
attribute), 19

NOT_IMPLEMENTED (cry-
tic_compile.platform.types.Type attribute),
12

notice (cryptic_compile.utils.natspec.UserDoc at-
tribute), 16

notice (cryptic_compile.utils.natspec.UserMethod at-
tribute), 16

O

optimized (cryptic_compile.compiler.compiler.CompilerVersion
attribute), 1

P

package (cryptic_compile.cryptic_compile.CrypticCompile
attribute), 19

package_name (cry-
tic_compile.cryptic_compile.CrypticCompile
attribute), 19

params (crytic_compile.utils.natspec.DevMethod attribute), 15
 platform(crytic_compile.crytic_compile.CryticCompile attribute), 20
 platform_name_used (crytic_compile.platform.abstract_platform.AbstractPlatform attribute), 3
 platform_name_used (crytic_compile.platform.standard.Standard attribute), 10
 platform_project_url_used (crytic_compile.platform.abstract_platform.AbstractPlatform attribute), 3
 platform_project_url_used (crytic_compile.platform.standard.Standard attribute), 10
 platform_type_used (crytic_compile.platform.abstract_platform.AbstractPlatform attribute), 3
 platform_type_used (crytic_compile.platform.standard.Standard attribute), 10
 PROJECT_URL (crytic_compile.platform.abstract_platform.AbstractPlatform attribute), 2
 PROJECT_URL (crytic_compile.platform.archive.Archive attribute), 3
 PROJECT_URL (crytic_compile.platform.brownie.Brownie attribute), 4
 PROJECT_URL (crytic_compile.platform.dapp.Dapp attribute), 5
 PROJECT_URL (crytic_compile.platform.embark.Embark attribute), 6
 PROJECT_URL (crytic_compile.platform.etherlime.Etherlime attribute), 6
 PROJECT_URL (crytic_compile.platform.etherscan.Etherscan attribute), 7
 PROJECT_URL (crytic_compile.platform.solc.Solc attribute), 8
 PROJECT_URL (crytic_compile.platform.solc_standard_json.SolcStandardJson attribute), 9
 PROJECT_URL (crytic_compile.platform.standard.Standard attribute), 10
 PROJECT_URL (crytic_compile.platform.truffle.Truffle attribute), 11
 PROJECT_URL (crytic_compile.platform.vyper.Vyper attribute), 12
 PROJECT_URL (crytic_compile.platform.waffle.Waffle attribute), 13

R

relative (crytic_compile.utils.naming.Filename attribute), 13
 relative_filename_from_absolute_filename (crytic_compile.crytic_compile.CryticCompile attribute), 20

S

save_to_zip() (in module crytic_compile.utils.zip), 17
 short (crytic_compile.utils.naming.Filename attribute), 14
 Solc (class in crytic_compile.platform.solc), 8
 SOLC (crytic_compile.platform.types.Type attribute), 12
 SOLC_STANDARD_JSON (crytic_compile.platform.types.Type attribute), 12
 SolcStandardJson (class in crytic_compile.platform.solc_standard_json), 9
 src_content_for_file() (crytic_compile.crytic_compile.CryticCompile attribute), 20
 srcmap_runtime() (crytic_compile.crytic_compile.CryticCompile attribute), 20
 srcmaps_init (crytic_compile.crytic_compile.CryticCompile attribute), 20
 srcmaps_runtime (crytic_compile.crytic_compile.CryticCompile attribute), 20
 Standard (class in crytic_compile.platform.standard), 9
 STANDARD (crytic_compile.platform.types.Type attribute), 12
 target (crytic_compile.crytic_compile.CryticCompile attribute), 20
 target (crytic_compile.platform.abstract_platform.AbstractPlatform attribute), 3
 title (crytic_compile.utils.natspec.DevDoc attribute), 15
 to_dict() (crytic_compile.platform.solc_standard_json.SolcStandardJson attribute), 9
 Truffle (class in crytic_compile.platform.truffle), 11
 TRUFFLE (crytic_compile.platform.types.Type attribute), 12
 Type (class in crytic_compile.platform.types), 11
 type (crytic_compile.crytic_compile.CryticCompile attribute), 20

TYPE (*crytic_compile.platform.abstract_platform.AbstractPlatform*
 attribute), 2
 TYPE (*crytic_compile.platform.archive.Archive* *at-*
 tribute), 3
 TYPE (*crytic_compile.platform.brownie.Brownie* *at-*
 tribute), 4
 TYPE (*crytic_compile.platform.dapp.Dapp* *attribute*), 5
 TYPE (*crytic_compile.platform.embark.Embark* *at-*
 tribute), 6
 TYPE (*crytic_compile.platform.etherlime.Etherlime* *at-*
 tribute), 6
 TYPE (*crytic_compile.platform.etherscan.Etherscan* *at-*
 tribute), 7
 TYPE (*crytic_compile.platform.solc.Solc* *attribute*), 8
 TYPE (*crytic_compile.platform.solc_standard_json.SolcStandardJson*
 attribute), 9
 TYPE (*crytic_compile.platform.standard.Standard*
 attribute), 10
 TYPE (*crytic_compile.platform.truffle.Truffle* *attribute*),
 11
 TYPE (*crytic_compile.platform.vyper.Vyper* *attribute*), 12
 TYPE (*crytic_compile.platform.waffle.Waffle* *attribute*),
 13

U

used (*crytic_compile.utils.naming.Filename* *attribute*),
 14
 used_filename_of_contract() (*cry-*
 tic_compile.crytic_compile.CryticCompile
 method), 20
 UserDoc (*class in crytic_compile.utils.natspec*), 15
 userdoc (*crytic_compile.utils.natspec.Natspec* *at-*
 tribute), 15
 UserMethod (*class in crytic_compile.utils.natspec*), 16

V

version (*crytic_compile.compiler.compiler.CompilerVersion*
 attribute), 1
 Vyper (*class in crytic_compile.platform.vyper*), 12
 VYPER (*crytic_compile.platform.types.Type* *attribute*), 12

W

Waffle (*class in crytic_compile.platform.waffle*), 13
 WAFFLE (*crytic_compile.platform.types.Type* *attribute*),
 12
 working_dir (*crytic_compile.crytic_compile.CryticCompile*
 attribute), 20