↑ Compact language Tools Compact Compiler Manual Page

Compact Compiler Manual Page

NAME

compactc

OVERVIEW

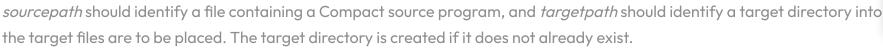
The Compact compiler, **compactc**, takes as input a Compact source program in a specified source file and translates it into several target files in a specified directory.

SYNOPSYS

compactc flag ... sourcepath targetpath

DESCRIPTION

The flags *flag* ... are optional. They are described under FLAGS later in this document.





Feedback

compactc compiles the source file and produces from it the following target files, where *sourceroot* is the name of the file identified by *sourcepath* without any extension.

- a Typescript type-definition file targetdir/contract/index.d.cts
- a Javascript source file targetdir/contract/index.cjs
- a Javascript source-map file targetdir/contract/index.cjs.map
- one Zk/ir circuit file for each exported circuit circuitname in targetdir/zkir/circuitname.zkir, and
- a pair of proving keys for each exported circuit *circuitname* in *targetdir*/keys/*circuitname*.prover and *targetdir*/keys/*circuitname*.verifier.

Compact source files can include other Compact source files via an include form:

include 'name';

They can also import externally defined modules via an **import** form:

import name; import 'name';

By default, the compiler looks for include files and externally defined modules in the current working directory under the full filename name.compact. When the environment variable COMPACT_PATH is set to a colon-separated (semicolon-separated on Windows) list of directory pathnames dirpath:...:dirpath (dirpath;...;dirpath under Windows), the compiler looks instead under the full pathname dirpath/name.compact for each dirpath until the file is found or the set of dirpath entries is exhausted.

Every Compact source program should import the standard library **CompactStandardLibrary**. This is typically done by placing the following line at the top of the program:

import CompactStandardLibrary;

FLAGS

The following flags, if present, affect the compiler's behavior as follows:

--help

prints help text and exits.

--version

prints the compiler version and exits.

--language-version

prints the language version and exits.

--vscode

causes the compiler to omit newlines from error messages, so that they are rendered properly within the VS Code extension for Compact.

--skip-zk

causes the compiler to skip the generation of proving keys. Generating proving keys can be time-consuming, so this option is useful when debugging only the Typescript output. The compiler also skips, after printing a warning message, the generation of proving keys when it cannot find zkir.

--no-communications-commitment

omits the contract communications commitment that enables data integrity for contract-to-contract calls.

--sourceRoot sourceRoot-value

overrides the compiler's setting of the sourceRoot field in the generated source-map (.cjs.map) file. By default, the compiler tries to determine a useful value based on the source and target-directory pathnames, but this value might not be appropriate for the deployed structure of the application.

--trace-passes

causes the compiler to print tracing information that is generally useful only to compiler developers.

EXAMPLES

Assuming **src/test.compact** contains a well-formed Compact program exporting circuits *foo* and *bar*:

```
compactc src/test.compact obj/test
```

produces:

```
obj/test/contract/index.d.cts
obj/test/contract/index.cjs
obj/test/contract/index.cjs.map

obj/test/zkir/foo.zkir
obj/test/zkir/bar.zkir

obj/test/keys/foo.prover
obj/test/keys/foo.verifier
obj/test/keys/bar.prover
obj/test/keys/bar.verifier
```

produces the same, except without the keys.