NAME

thrift — compiler for Thrift IDL language

SYNOPSIS

thrift [**-version**] [**-o** *dir*] [**-out** *dir*] [**-I** *dir*] [**-nowarn**] [**-strict**] [**-v**[erbose]] [**-r**[ecurse]] [**-debug**] [**--allow-neg-keys**] [**--allow-64bit-consts**] [**--gen** *generator-string*] *file*

DESCRIPTION

thrift is a compiler that generates source code. It reads the Thrift IDL language and generates code that marshals the defined data structures across a TCP/IP network. **thrift** generates code for a great many languages, see **GENERATORS** below.

The options are:

-o *dir* Set the output directory for gen-∗ packages (default: current directory)

-out *dir* Set the ouput location for generated files (no gen-∗ folder will be created)

-I *dir* Add a directory to the list of directories searched for include directives

-nowarn Suppress all compiler warnings

-strict Strict compiler warnings on

-v[erbose] Verbose mode

-r[ecurse] Also generate included files

-debug Parse debug trace to stdout

-version Print the compiler version

--allow-neg-keys
Allow negative field keys (Used to preserve protocol compatibility with older .thrift files)

--allow-64bit-consts
Do not print warnings about using 64-bit constants

--gen *generator-string*
Generate code with a dynamically-registered generator. *generator-string* has the form
language[:*key*[=*value*[,*key*[=*value*...]]]]
key and *value* are options passed to the generator. Many options do not require *value*. Whitespace is not permitted in *generator-string*.

GENERATORS

Available generators and options:

as3 (AS3)
bindable
Add *bindable* metadata to all the struct classes.

c_glib (C, using GLib)

cocoa (Cocoa)

log_unexpected
Log every time an unexpected field ID or type is encountered.

cpp (C++)

cob_style
Generate “Continuation OBject”-style classes.

no_client_completion
Omit calls to *completion__()* in *CobClient* class.

templates
Generate templatized reader/writer methods.

pure_enums
Generate pure enums instead of wrapper classes.

dense Generate type specifications for the dense protocol.

include_prefix
Use full include paths in generated files.

csharp (C#)

async Adds Async CTP support.

wcf Adds bindings for WCF to generated classes.

d (D)

delphi (delphi)

ansistr_binary
Use AnsiString as binary properties.

erl (Erlang)

go (Go)

hs (Haskell)

html (HTML)

java (Java)

beans Members will be private, and setter methods will return void.

private-members
Members will be private, but setter methods will return ’this’ like usual.

nocamel
Do not use CamelCase field accessors with beans.

hashcode
Generate quality hashCode methods.

android_legacy
Do not use *java.io.IOException(throwable)* (available for Android 2.3 and above).

java5 Generate Java 1.5 compliant code (includes android_legacy flag).

javame (Java ME)

js (Javascript)

jquery Generate jQuery-compatible code.

node Generate node.js-compatible code.

ocaml (OCaml)

perl (Perl)

php (PHP)

inlined
: Generate PHP inlined files.

server Generate PHP server stubs.

oop Generate PHP with object oriented subclasses.

rest Generate PHP REST processors.

py (Python)

new_style
: Generate new-style classes.

twisted
: Generate Twisted-friendly RPC services.

utf8strings
: Encode/decode strings using utf8 in the generated code.

slots Generate code using slots for instance members.

dynamic
: Generate dynamic code, less code generated but slower.

dynbase=class
: Derive generated classes from *class* instead of *TBase*.

dynexc=class
: Derive generated exceptions from *class* instead of *TExceptionBase*.

dynimport='from `filename` import *class*'
: Add an import line to generated code to find the dynbase *class*.

rb (Ruby)

rubygems
: Add a `require 'rubygems'` line to the top of each generated file.

st (Smalltalk)

xsd (XSD)

ENVIRONMENT

none

FILES

none

EXAMPLES

none

DIAGNOSTICS

thrift exits 0 on success, and >0 if an error occurs. (It is to be hoped.)

BUGS

May your shepherd eat your goats.