NAME

omniidl − omniORB idl compiler

SYNOPSIS

omniidl [options] −b<back-end> [back-end options] file

DESCRIPTION

omniidl is the omniORB IDL compiler front end. If a back-end is not specified, it checks the input IDL file for validity, and produces no output. Usually, a language mapping back-end is specified, so stubs and skeletons in the target language are produced.

The input files are processed by the C preprocessor before they are parsed by the compiler.

COMMON OPTIONS

−b<backend> Run the specified back-end (e.g., −bcxx = C++, −bpython = Python)

−D<name>=<value>
Define <name> for the C preprocessor.

−U<name> Undefine <name> for the C preprocessor.

−I<dir> Include <dir> in the C preprocessor search path.

−E Only run the C preprocessor, sending its output to stdout.

−Y<cmd> Use <cmd> as the preprocessor instead of the default.

−N Do not run the C preprocessor.

−Wp<arg> Send <arg> to the C preprocessor.

−Wb<arg> Send <arg> to the back-end.

−nf Do not warn about unresolved forward declarations.

−k Keep comments after declarations, to be used by some back-ends.

−K Keep comments before declarations, to be used by some back-ends.

−C<dir> Change directory to <dir> before writing output files.

−d Dump the parsed IDL then exit, without running a back-end.

−p<dir> Use <dir> as a path to find omniidl back-ends.

−V Print version information then exit.

−u Print usage information.

−v Verbose: trace compilation stages.

C++ BACK-END

Choose the C++ back-end with **−bcxx.** The C++ back-end is only available when you have omniORB for C++ installed.

The C++ back-end produces two output files: a header and a stub/skeleton file. By default they are named by appending suffixes **.hh** and **SK.cc** to the base name of the input IDL file.

If the **−Wba** option is specified, then a third file is generated (with default suffix **DynSK.cc**), containing code for TypeCode and Any.

C++ BACK-END OPTIONS

−Wbh=<suffix> Use <suffix> instead of .hh

−Wbs=<suffix> Use <suffix> instead of SK.cc

−Wbd=<suffix> Use <suffix> instead of DynSK.cc. If the same suffix is specified for **−Wbs** and **−Wbd** then a single skeleton file containing all the definitions is output.

−Wba Generate definitions for TypeCode and Any.

−Wbinline Output stubs for #included IDL files in line with the main file.

−Wbtp Generate tie implementation skeletons.

−Wbtf Generate flattened tie implementation skeletons.

−Wbsplice-modules
Splice together multiply-opened modules into one.

−Wbexample Generate example implementation code.

−WbBOA Generate BOA compatible skeletons.

−Wbkeep_inc_path
Preserve IDL #include paths in generated #include directives.

−Wbuse_quotes Use quotes in #include directives (e.g. "foo" rather than <foo>).

PYTHON BACK-END

Choose the Python back-end with **−bpython.** The Python back-end produces Python packages according to the standard IDL to Python mapping, to be used with omniORBpy. The Python back-end is only available when you have omniORBpy installed.

The Python back-end generates Python package directories named after the modules declared in IDL, as required by the IDL to Python mapping. It also creates separate stub files that are imported by the packages.

PYTHON BACK-END OPTIONS

−Wbinline Output stubs for #included IDL files in line with the main file.

−Wbglobal=<name>
Use <name> as the name for the global IDL scope (default _GlobalIDL).

−Wbpackage=<name>
Put both Python modules and stub files in package <name>.

−Wbmodules=<name>
Put Python modules in package <name>

−Wbstubs=<name>
Put stub files in package <name>

SEE ALSO

See the omniORB or omniORBpy manual for full details of **omniidl.**

AUTHOR

Duncan Grisby