User manual

COM2CORBA

Date: 2009-05-31
Version: 1.1.0 (revision 442)

This document

Summary A manual describing usage from a COM2CORBA tool user perspective.

Author Joel Purra

Date 2009-05-31

Version 1.1.0 (revision 442)

Document history

Version	Date	Changes
0.1.0	2009-05-07	Document created.
1.0.0	2009-05-08	First version
1.1.0	2009-05-31	Added a few troubleshooting tips

Contents

Chapter 1

Tool description

COM2CORBA is a tool to help generate communication code between your COM client program and a CORBA service. This normally requires a lot of manual coding, checking and type-casting from and to COM and CORBA types. The intention of this tool is to simplify these steps to one step that, according to a given CORBA interface, generates all layers needed.

Chapter 2

Prerequisites

- The compiled COM2CORBA tool
- Microsoft Visual Studio

Chapter 3

Usage

Always start the COM2CORBA generator tool through a Visual Studio command prompt as it sets the necessary environment variables.

3.1 Basic usage

To generate a new file, the basic syntax is

```
com2corba "path to\some.idl"
```

To define an output directory that isn't the tool's working directory, use the -o "folder path" flag

```
com2corba -o "other\output folder" "path to\some.idl"
```

TAO's IDL compiler invokes the C/C++ preprocessor to resolve IDL files, which means preprocessor flags such as -D and -I can be used.

```
com2corba -DDEBUG "path to\some.idl"
```

3.2 Reference

COM2CORBA passes all arguments to the TAO compiler, which means any TAO command line arguments could be used. Please refer to the TAO options documentation[1] for further information. The Options for TAO Components document can you find as a copy in the 'User manual tao options' document.

[1] Options for TAO Components `http://www.dre.vanderbilt.edu/~schmidt/DOC_ROOT/TAO/docs/compiler.html`

Chapter 4

Input files

4.1 IDL-file

For the tool to know what to generate, an IDL file that describes the CORBA service is required. The service supplier should have this prepared and ready for you to use.

Optional areas, for example for debugging reasons, and include files may be used through C/C++ preprocessor commands.

4.2 ODL-file

The COM version of an IDL file, used by COM2CORBA to keep COM GUID mappings between tool runs. This helps keep your COM program in line as references are kept intact.

The name of the ODL file is assumed to be the same as the input IDL file but with the .odl file extension. Not necessary to supply, as it will be generated and/or modified when the tool is used to generate code.

Chapter 5

Output

The tool outputs ready-to-use C++ COM wrapper code for the TAO generated C++ CORBA wrapper.

5.1 Files

These are the files generated for an input file with the name "InputFile.idl". Include the output files in your Visual Studio C++ COM project.

5.1.1 Generated by COM2CORBA

- InputFile.odl
- InputFiledll.cpp
- InputFiledll.h
- InputFile_aut.h
- InputFile_aut00.cpp
- InputFile_aut01.cpp
- InputFile_bo.h
- InputFile_bo00.cpp
- InputFile_bo01.cpp
- InputFile_debug.cpp
- stdafx.h

5.1.2 Generated by TAO

- InputFileC.cpp
- InputFileC.h
- InputFileC.inl
- InputFileS.cpp

- InputFileS.h
- InputFileS.inl

5.2 Usage

5.3 Compilation

5.3.1 Referenced by TAO

Chapter 6

Troubleshooting

6.1 com2corba.exe: spawn of "CL.EXE" failed

Make sure COM2CORBA is started in a Visual Studio Command Prompt, or through Visual Studio itself after compilation.

6.2 When I run the tool through Visual Studio after compilation, nothing happens

Right click on the COM2CORBA project and select properties. Under Configuration Properties, Debugging, change Command Arguments to fit your needs. The basic case is to just enter the name of an IDL file in the working directory of COM2CORBA.

6.3 The COM2CORBA window closes too fast when I run it through Visual Studio

Either run the tool manually in a separate window or propose the following to your COM2CORBA developer/maintainer. In the TAO_IDL folder of ACE+TAO, edit the last lines of tao_idl.cpp to match the lines below. This way the window won't close until you hit enter. You can also hit enter before reaching this final line and make the window close as soon as the program is done. This requires a recompilation to come in effect; see the installation manual.

```
int retval = idl_global->err_count ();
DRV_cleanup ();
// Don't close window
std::cin.get();
return retval;
```

Changing ACE+TAO isn't really recommended as changes breaks when ACE+TAO gets updated.