CANTATA

Technical Note

Use with MSVC Debugger

Issue 1.0

The purpose of this Technical Note is to outline a method that can be used to help debugging a Cantata test script within MSVC, when Cantata makefiles are being used. This document will focus on using Microsoft Visual Studio 2010 however, the same principals can be used with any version of MSVC to debug a Cantata Test Script.

Contents

Copyright Notice

1 Introduction

The purpose of this Technical Note is to outline a method that can be used to help debugging a Cantata test script within MSVC, when Cantata makefiles are being used. This document will focus on using Microsoft Visual Studio 2010 however, the same principals can be used with any version of MSVC to debug a Cantata Test Script.

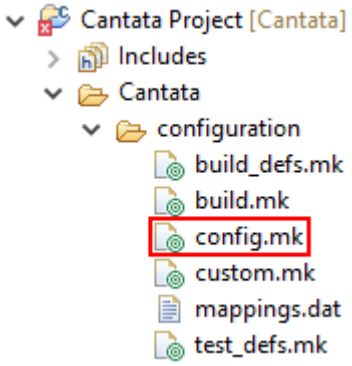
Changes will need to be made to the Cantata makefiles, the test script itself and the ipg.cop.

The idea behind this is that we will insert a pause into the test script and then attach Visual Studios debugger to the test process while it is running. To do this we need to allow the debug options for Visual Studio and keep the modified file so that we are able to step through it.

2 Setting up for Debugging

2.1 Make Files

Open the **config.mk** file within the Cantata, configuration directory.



And add **/DEBUG** to the **ORIGINAL_DEFAULT_LINKER_OPTS** and the **DEFAULT_LINKER_OPTS**.

And **/Zi** to the **ORIGINAL_DEFAULT_COMPILER_OPTS** and the **DEFAULT_COMPILER_OPTS**.

```
#
# Compiler specific options
#
DEFAULT_COMPILER_NAME := cl.exe
ifeq ($(USE_ESCAPED_SLASHES),1)
    ORIGINAL_DEFAULT_COMPILER_OPTS := -TP -c %f -Fo%o /Zi
    DEFAULT_COMPILER_OPTS := $(patsubst /%,//%,$(ORIGINAL_DEFAULT_COMPILER_OPTS))
    ORIGINAL_DEFAULT_LINKER_OPTS := /out:%o %f /DEBUG
    DEFAULT_LINKER_OPTS := $(patsubst /%,//%,$(ORIGINAL_DEFAULT_LINKER_OPTS))
    ORIGINAL_DEFINE_OPT := /D
    DEFINE_OPT := $(patsubst /%,//%,$(ORIGINAL_DEFINE_OPT))
    ORIGINAL_LIBRARY_OPT := /DEFAULTLIB:%s
    LIBRARY_OPT := $(patsubst /%,//%,$(ORIGINAL_LIBRARY_OPT))
    ORIGINAL_INCLUDE_PATH_OPT := /I%s
    INCLUDE_PATH_OPT := $(patsubst /%,//%,$(ORIGINAL_INCLUDE_PATH_OPT))
else
    DEFAULT_COMPILER_OPTS := -TP -c %f -Fo%o /Zi
    DEFAULT_LINKER_OPTS := /DEBUG /out:%o %f
    DEFINE_OPT := /D
    LIBRARY_OPT := /DEFAULTLIB:%s
    INCLUDE_PATH_OPT := /I%s
endif
DEFAULT_LINKER_NAME := link.exe
OBJ_FILE_EXTS := .obj
MULTIPLEXED_OUTPUT := 0
```

2.2 Cantata Test Script

You need to add a pause to the test script. To do this open the test script in the C/C++ Editor. At the start of the **main** function add:

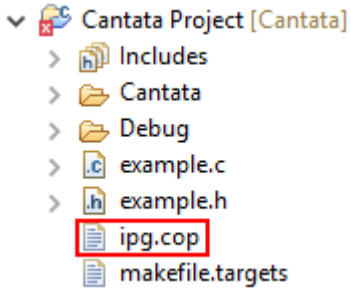
```
int vl_int;
        printf ("Pausing for debug purposes – enter a number\n");
        scanf ("%d", &vl_int);
```

This will pause the test once it starts running. The test script will look something like this:

```
/*****************************************************************************/
/* Program Entry Point                                                       */
/*****************************************************************************/
int main()
{
    int vl_int;
       printf ("Pausing for debug purposes - enter a number\n");
       scanf ("%d", &vl_int);

    OPEN_LOG("test_reverse_string.ctr", false, 100);
    START_SCRIPT("reverse_string", true);

    TEST_CLASS(reverse_string) test_object;
    test_object.run_tests();

    return !END_SCRIPT(true);
}
```

2.3 ipg.cop

Open the project level ipg.cop file:

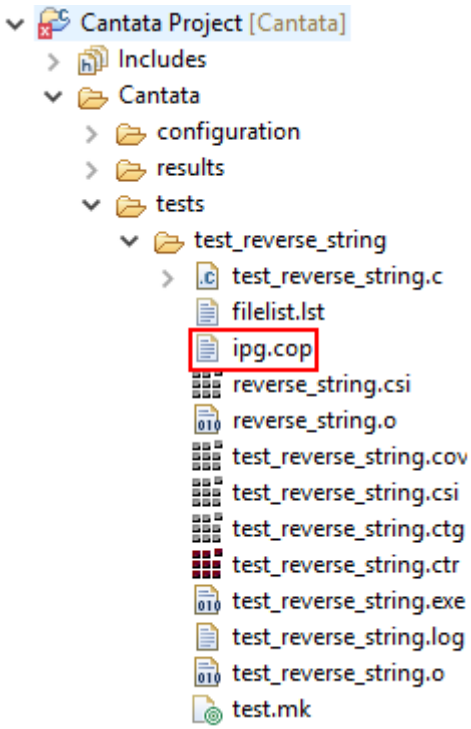


Add **--keepmod** to the User Section, this will make sure the pre-compiled files are kept when the test script is built.

Also change **--parse:--line_directives** to **--parse:--no_line_directives** this will prevent #line directives from being output in the modified source file.

```
#
# Cantata Project-Level Options
#
# The options set in this file will be inherited by each test in the project.
#
# WARNING: Do not alter this file manually.
#
#tool.use=true
#tool.tests=Cantata
#tool.version=2
"--analyse"
"--parse:--no_line_directives"
"--parse:-W2"
"--sm:--call_seq_code"
"--comp:x86-Win32-5.x-msvc9"

#User Section
--keepmod
```

In the test level ipg.cop file:



Comment out the line **"--ci:--instr:stmt;func;call;decn;"** this can be done by adding a **#** to the start of the line.

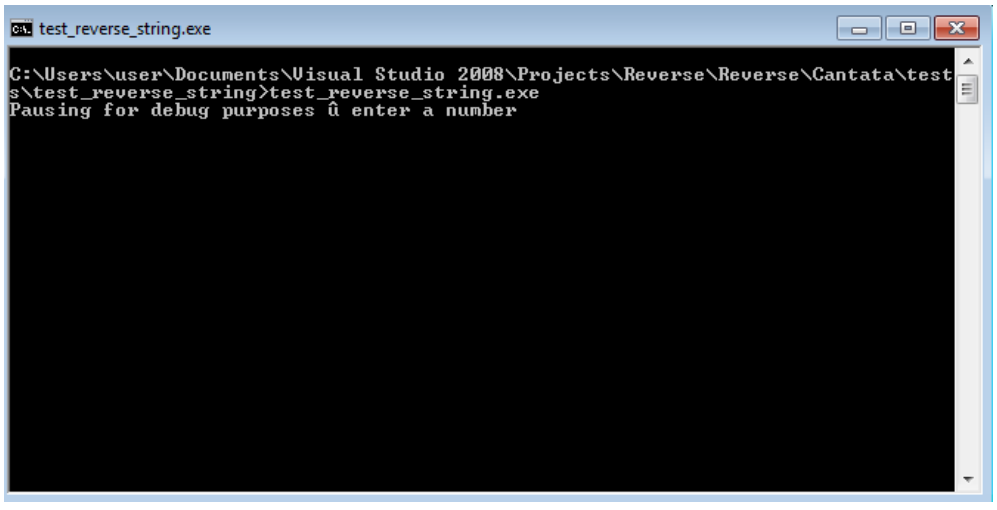
```
#
# Cantata Test-Level Options
#
# Any options set in this file will override the options set in the
# project-level options file.
#
# WARNING: Do not alter this file manually.
#
#"--ci:--instr:stmt;func;call;decn;"
"--sm:--wrap:strlen(const char*)#reverse_string(char *)"
"--sm:--wrap:operator new[](size_t)#reverse_string(char *)"

#User Section
```

The reason we remove the coverage is for two reasons. Firstly, it should not need debugging as it should not be causing an error or issue within the test script. Secondly, if it is not removed then there will be a call to a Cantata library for every function/check, meaning that it will take twice as long if it is left in.

3 Running the Debugging

3.1 Running the Test

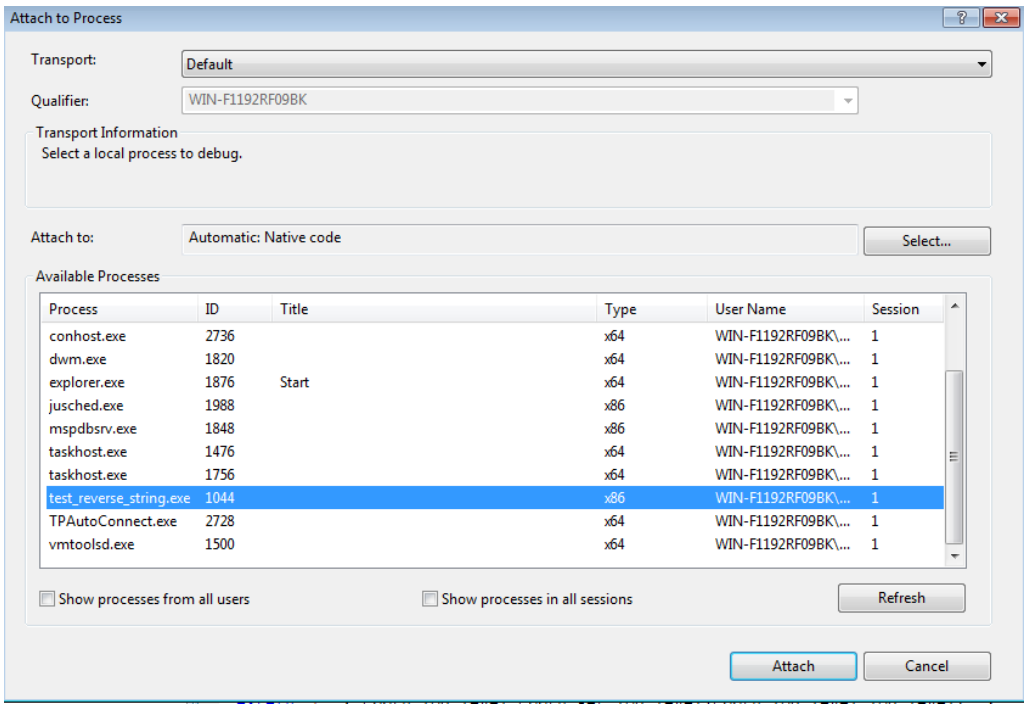
Once the above changes have been made, build the test script with the default build configuration set to Cantata. You do not want to run the test from within Cantata as it does not allow input. Instead, open a command prompt and navigate to the Cantata test directory for the project and run the test.exe. It should start the test and then pause awaiting a user input.



3.2 Attaching the MSVC Debugger

Open Visual Studio and go to **Debug>Attach to Process**.

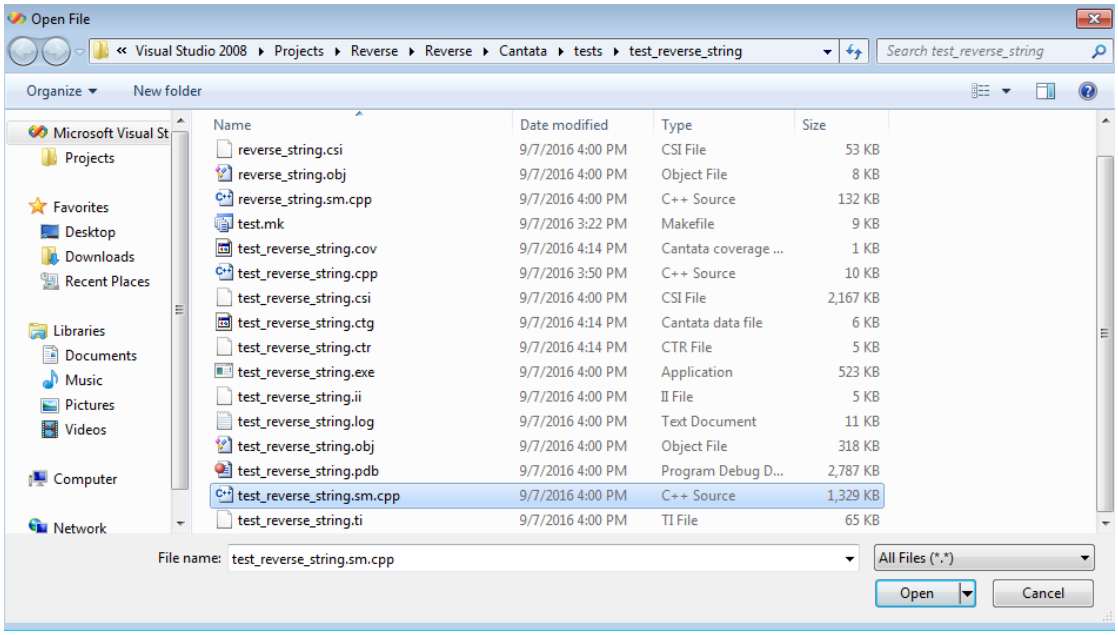
This will bring up a list of processes that you could debug, select the test.exe you have running and select **Attach**.



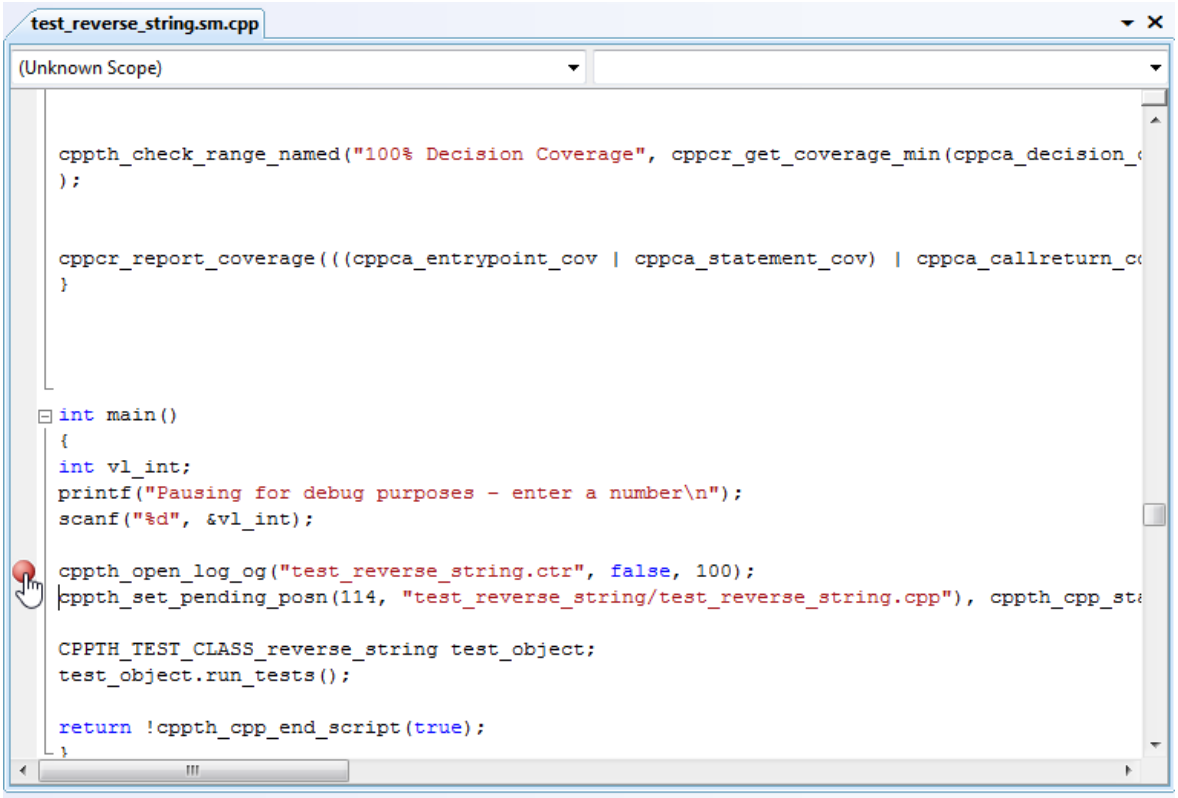
Next, go to **File>Open>File…** (This is still in Visual Studio)

Navigate to the Cantata test directory and the test script you want to debug.

Select the **.sm.cpp** file that has been generated, this is most likely the one closest to the bottom if you are sorting by name, and open it.



Find your main entry point in this file. And add a breakpoint by click in the margin on the left-hand side.



3.3 Debugging

Then go back to the command prompt. Type **1** and press **Enter**.

Back in Visual Studio, you should now be able to step through the test script and discover where/what is causing the error you are seeing.