Configuring Testing with the Cross Compiler

This topic explains how to add a definition for your cross-compiler to C++test so that you can use it in all projects which you normally compile with this compiler.

Sections include:

Understanding When to Add a Custom Compiler Definition

In general, a custom compiler should be defined when:

- Your compiler / linker names differ from the default ones for a supported compiler type (e.g. for a custom build of a gcc compiler).
- You are using an unsupported compiler which is option-compatible with a supported compiler type (e.g., some versions of ARM compilers or QCC from QNX).
- Linking requires fixed options that reference specific libraries (e.g., for embedded projects).

By defining a custom compiler, you can modify compiler/linker names and the specific patterns for how C++test uses the compiler and linker.

Adding a Custom Compiler Definition

Different approaches to adding custom compiler definitions are recommended for different C++test installation scenarios:

- **For installations on mounted file systems, when multiple users refer to the same C++test installation (typical on UNIX systems):** The custom compiler definition should be added by a user with write permissions to the C++test installation directory. After that, the created custom compiler definitions can be utilized by all C++test users.
- **For local installations (typical for Windows):** The custom compiler definition files need to be created with the wizard (as defined below) and then copied to a shared location (or, even better, checked into source control as utility files). Each user should then import these definitions into his own local installation using the Import functionality described in Importing Custom Compiler Definitions

Once you have a new definition of your custom compiler, you can use it in all projects that you normally compile with this compiler. When we discuss "building the test harness for target platform," it is assumed that you are using the suitable compiler definition. This is a part of defining the test flow.

To add a custom compiler definition to C++test.

1. (Optional) If you want to specify the location of the custom compiler definition files, choose **Parasoft> Preferences**, select **Parasoft> Configurations**, then enable **Custom directories> Custom compilers** and enter the location in the appropriate field .
    - By default, the custom compiler definition files are saved in `<workspace>\metadata\.plugins\com.parasoft.xtest.checkers.eclipse.core.cpp/compilers`.
2. Choose **File> New> Other**, choose **C++test> Custom compiler**, then click **Next**.The New Custom Compiler dialog will open.
3. Select **Add custom compiler**, then click **Next**.
4. In the next page, specify the following custom compiler settings:
    - **Compiler name:** The unique name that will be used to identify this custom compiler in the C++test GUI.
    - **Compiler family:** The family of compilers which corresponds to your actual compiler (if you are not sure, choose one of the GCC compilers).
    - **Compiler identifier:** The unique name that will be used to identify the directory in which its configuration settings are stored. This name should conform to all limitations that your OS file system imposes on directory names.
    - **C compiler executable:** The C compiler executable.
    - **C++ compiler executable:** The C++ compiler executable.
    - **Linker executable:** The linker executable. The compiler and linker settings must be consistent.
5. (Optional) If you want to see the paths to configuration files created for your new custom compiler, click **Next.**
6. Click **Finish.**

When a new compiler definition is added, C++test creates a set of configuration files which can be customized. These files include:

- **Compiler definition file:** This file stores details about the configuration of compiler executables and command line patterns which should be used when compiling the test harness.

> **Important**
> When dealing with a cross compiler, most users typically need to alter the linker command line by removing the references to the following precompiled version of the C++test runtime library that is used in the linker command lines by default:
> `“ $(CPPTEST_LIB_DIR)/cpptestruntime.lib“`
> They should be replaced with references to the custom, cross-compiled version of the runtime library. For example:
> `“linkerCmdLine=$(exe) $(filtered_opts) $(input) $(HOME)/cpptest/ppc603/cpptestruntime.a -o $(output)”`
> For details on how to build runtime library, see Working with the C++test Runtime Library.

- **Configuration for C file:** This file is `c.psrc`.
- **Configuration for C++ file:** This file is `cpp.psrc`.

Importing Custom Compiler Definitions

Any custom compiler definitions that are available in the designated Custom Compilers location will be loaded automatically upon startup, and available for use. Thus, one way to "import" definitions is to save them in the Custom Compiler location. By default, the custom compiler definition files are saved in<workspace>\.metadata\.plugins\com.parasoft.xtest.checkers.eclipse.core.cpp/compilers.To change this location, choose **Parasoft> Preferences**, select **Parasoft> Configurations**, then enable **Custom directories> Custom compilers** and enter the location in the appropriate field .

To import a custom compiler definition that is stored in another location:

1. Choose **File> New> Other**, choose **C++test> Custom compiler**, then click **Next.** The New Custom Compiler dialog will open.
2. Select **Import custom compiler**, then click **Next**.
3. In the **Base compiler definition directory** field, specify the directory which contains the compiler definition files (described in the previous section) that you want to import.
    - Ensure that the name of the imported compiler directory does not overlap with the name of any existing configurations.
4. Click **Finish**.

Setting Build Options

After adding a custom compiler, ensure that your Project Options' Build Settings are set properly:

1. Load (or create) the C++test project in the GUI.
    - For details, see Creating a Project.
2. In the project tree, right-click the project node, then choose **Properties** from the shortcut menu.
3. In the Properties panel, expand the **Parasoft> C++test** category and select **Build Settings**.
4. From the **Compiler settings> Family** box, select the name of your custom compiler.
5. Add the library that you built with the custom compiler to the linker options by changing the Linker options. For example, you might change it from
   `${cpptest:original_options}`
   to
   `${cpptest:original_options} "C:\cygwin\home\<user_name>\Custom_Compiler\source\target\libcpptestruntime.a"`
6. Click **Apply**, then **OK**.

Tips

- The compiler family setting should correspond to the name of the actual compiler.
- The compiler and linker settings should be set for your environment, and should be consistent.
- For Managed Make projects, compiler and linker settings are specified in the managed project settings.
- See Setting Project and File Options for details on reviewing and setting project options.