Unit Testing with the C/C++test Wind River Workbench Plugin

This topic explains how you can execute unit tests with C/C++test Plugin for Wind River Workbench 4.x. See Test Creation and Execution for general information about running unit tests with C/C++test.

The following steps are required to execute and collect information for unit tests with C/C++test Plugin for Wind River Workbench 4.x:

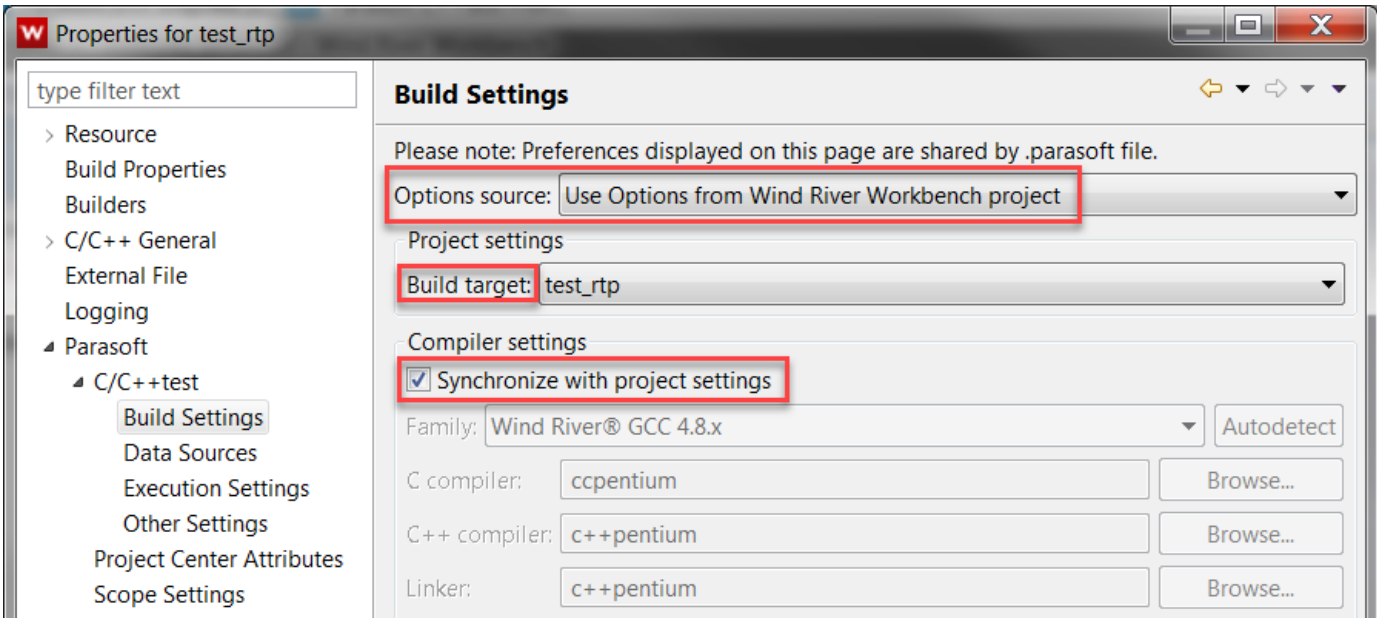
Configuring the Project

C/C++test Plugin for Wind River Workbench 4.x supports the following project types:

- Downloadable Kernel Module
- Real-Time Process

Before test execution, be sure that your project can be built in Wind River Workbench and is properly configured. Open the **Properties** of the project, go to **Parasoft> C/C++test> Build Settings** and ensure that the following options are configured:

- The **Use Options from Wind River Workbench project** option is selected from the **Option source** drop-down menu.
- The **Build target** option is properly set.
- The **Synchronize with project settings** option is enabled.



The compiler family and executables are automatically set during during the first analysis run.

Configuring the Execution Environment for Test Automation

Before test execution, be sure that your project can be successfully run in your execution environment (Simulator or Target) and is properly configured.

- Ensure that the Debug Agent is enabled in your VxWorks runtime environment.
- Ensure that the Debug Agent target address matches the target address specified in the test configuration.
  The default target address used by C/C++test is `127.0.0.1:60000`. You can customize the target address in the test configuration (see Customizing the Test Configuration).

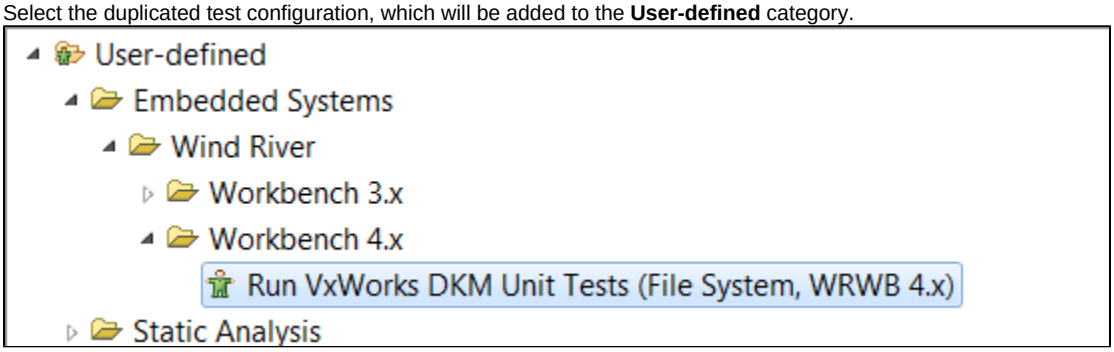
Avoid using network port auto-mapping for the Debug Agent. You can specify a fixed Debug Agent local port when configuring VxWorks Simulator. Go to **Advanced> Network Config> Configure...** and add a new Port Mapping for debug_agent: **Remote Port: 1534 > Local Port: 60000**.

- Ensure that the file system with the host-target path mapping (such as HostFS, PassFS) is available in your VxWorks runtime environment.
  By default, C/C++test uses the automatically detected host-target mapping when collecting test and coverage results. You can customize the mapping in the test configuration (see Customizing the Test Configuration).

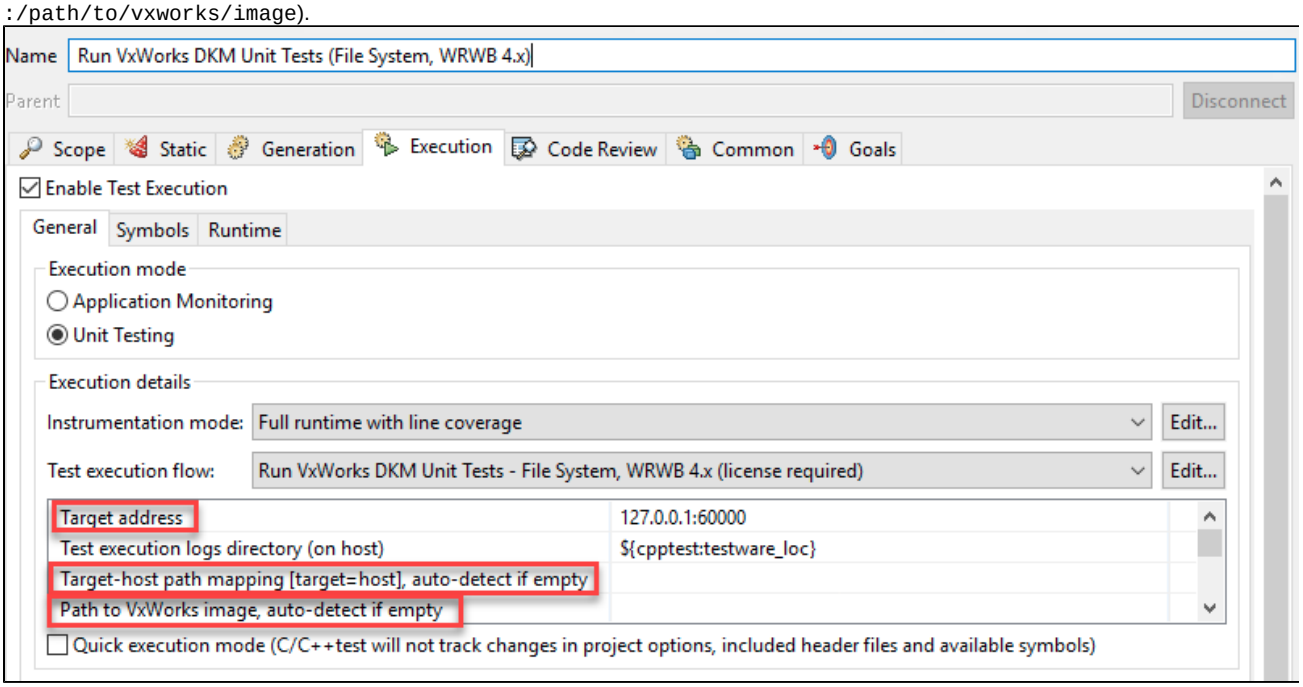
Customizing the Test Configuration

To review or modify the configuration settings for your execution environment:

1. Open **Parasoft> Test Configurations** in your IDE menu.
2. Go to **Builtin> Embedded Systems> Wind River> Workbench 4**.
3. Depending on your project type, right-click one of the following test configurations and choose **Duplicate**:
   - **Run VxWorks DKM Unit Tests (File System, WRWB 4.x)** for DKM projects
   - **Run VxWorks RTP Unit Tests (File System, WRWB 4.x)** for RTP projects.
4. Select the duplicated test configuration, which will be added to the **User-defined** category.



5. Go to **Execution> General> Execution** details and review or modify the following settings:
   - **Target address** - Specifies the target address (`host:port`); the default value is `127.0.0.1:60000`. Be sure the address matches the settings of your VxWorks Simulator or Running Target.
   - **Target-host path mapping** - Specifies path mapping between the target and the host (`/target/path=/host/path`). The mapping is used by C/C++test to store and access test and coverage log files. The mapping must include the host location specified in "Test execution logs directory (on host)". By default, C/C++test auto-detects and uses available mappings provided by the Debug Agent.
   - **Path to VxWorks image (DKM projects only)** - Specifies the path to the VxWorks image that is used to extract information about available symbols (required for correctly configuring stubs). If you use VxWorks Simulator, C/C++test automatically detects the VxWorks image. If you connect to Running Target, you must manually provide the path to the image. Be sure to use Unix-style path separators (c `:/path/to/vxworks/image`).



Creating Test Cases and Configuring Stubs

See Test Creation and Execution for instructions how to create and configure tests with C/C++test.

See Stubs for information about creating and configuring stubs.

> The newly added test files and stub files are automatically excluded from the build target.

Creating Stubs for DKM Projects

If you configure stubs for DKM projects, we strongly recommend collecting information about symbols that are available in the VxWorks kernel. You can achieve this by creating a custom test configuration:

1. Duplicate the **Builtin>Generate Stub**s configuration.
2. Rename the new configuration as **Generate Stubs for DKM Projects**.
3. Go to **Execution> General> Execution details> Test execution flow** and select **Generate Stubs for VxWorks DKM - WRWB 4.x** from the drop-down menu.
4. (Optional) Configure the path to the VxWorks image (see Customizing the Test Configuration).
5. Click **Apply**.

Executing Unit Tests

> Ensure your VxWorks Simulator or Running Target is connected.

1. Select the tests you want to execute.
2. Depending on your project type, run one of the following test configurations:
   - **Run VxWorks DKM Unit Tests (File System, WRWB 4.x)** for DKM projects
   - **Run VxWorks RTP Unit Tests (File System, WRWB 4.x)** for RTP projects.
   See Running a Test Configuration for information about running test configurations.

C/C++test will automatically:

- instrument the code
- build the test binary
- load the binary into the target (using the Debug Agent)
- execute tests (using the Debug Agent)
- unload the binary from the target (using Debug Agent)
- collect test and coverage results (using the Debug Agent / host-target path mapping)

Reviewing the Results

When the test execution is completed, you can review:

- test execution results; see Exploring Test Results for more details,
- coverage information collected during the test run; see Reviewing Coverage Information for details.