

IBM® Rational® Rhapsody® TestConductor Add On



Testing on a VxWorks Target

Rhapsody®

**IBM® Rational® Rhapsody®
TestConductor Add On**

Testing on a VxWorks Target

Release 2.8.0



License Agreement

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of the copyright owner, BTC Embedded Systems AG.

The information in this publication is subject to change without notice, and BTC Embedded Systems AG assumes no responsibility for any errors which may appear herein. No warranties, either expressed or implied, are made regarding Rhapsody software including documentation and its fitness for any particular purpose.

Trademarks

IBM[®] Rational[®] Rhapsody[®], IBM[®] Rational[®] Rhapsody[®] Automatic Test Generation Add On, and IBM[®] Rational[®] Rhapsody[®] TestConductor Add On are registered trademarks of IBM Corporation.

All other product or company names mentioned herein may be trademarks or registered trademarks of their respective owners.

© Copyright 2000-2017 BTC Embedded Systems AG. All rights reserved.

Contents

Content

Contents.....	4
Contacting IBM® Rational® Software Support.....	5
Introduction.....	6
Execution of TestCases on the VxWorks Target (animation based testing mode).....	7
Preparing the Code Generation Configuration.....	7
Preparing the Test Architecture.....	7
Executing a TestCase.....	8
Pitfalls and Limitations.....	9
Executing TestCases on the Target.....	9
Execution of TestCases on the VxWorks Target (assertion based testing mode).....	10
File IO needed for assertion based testing.....	10
Environment VxWorks6diab, VxWorks6gnu: Preparing the Code Generation Configuration.....	11
Properties of the Code Generation Configuration	11
Tags of the Code Generation Configuration:.....	11
Environment VxWorks6diab_RTP, VxWorks6gnu_RTP: Preparing the Code Generation Configuration.....	13
Properties of the Code Generation Configuration	13
Tags of the Code Generation Configuration:.....	13
Environment VxWorks6diab, VxWorks6diab_RTP, VxWorks6gnu, VxWorks6gnu_RTP: Executing a TestCase.....	15
Computation of code coverage.....	16
Target configuration.....	16
Options file for computation of code coverage (gnu compiler).....	16
Options file for computation of code coverage (diab compiler).....	18
Building and executing tests with computation of code coverage.....	20
Environment WorkbenchManaged653: Preparing the Code Generation Configuration.....	21
Settings of the Code Generation Configuration:.....	21
Properties of the Code Generation Configuration	21
Tags of the Code Generation Configuration:.....	22
Environment WorkbenchManaged653: Building and executing a TestCase.....	24
Environment WorkbenchManaged653: Computation of code coverage.....	25
Target configuration.....	25
Options file for computation of code coverage.....	25
Modify compile command in Workbench project.....	27
Avoid compile errors due to annotated code in tmp folder.....	29
Limitations.....	30
Support for VxWorks 653 (Rhapsody environment WorkbenchManaged653).....	30

Contacting IBM® Rational® Software Support

IBM Rational Software Support provides you with technical assistance. The IBM Rational Software Support Home page for Rational products can be found at <http://www.ibm.com/software/rational/support/>.

For contact information and guidelines or reference materials that you need for support, read the [IBM Software Support Handbook](#).

For Rational software product news, events, and other information, visit the [IBM Rational Software Web site](#).

Voice support is available to all current contract holders by dialing a telephone number in your country (where available). For specific country phone numbers, go to <http://www.ibm.com/planetwide>.

Before you contact IBM Rational Software Support, gather the background information that you will need to describe your problem. When describing a problem to an IBM software support specialist, be as specific as possible and include all relevant background information so that the specialist can help you solve the problem efficiently. To save time, know the answers to these questions:

What software versions were you running when the problem occurred?

Do you have logs, traces, or messages that are related to the problem?

Can you reproduce the problem? If so, what steps do you take to reproduce it?

Is there a workaround for the problem? If so, be prepared to describe the workaround.

Introduction

This document describes how TestCases can be executed with IBM® Rational® Rhapsody® TestConductor Add On on a VxWorks target, while Rhapsody is running on a Windows or Linux host. We assume the basic installation is already done: The tools needed to develop software for a VxWorks target are installed (for example Wind River WorkBench, compiler, VxWorks simulator, etc.). Rhapsody is installed on the Windows or Linux host including the add ons for development of applications for a VxWorks target, and a TCP/IP connection between the host and the WorkBench is available. Rhapsody running on the host will invoke the tested application on the target via this TCP/IP connection, and during test execution the communication between Rhapsody animation and the application uses TCP/IP. We will describe the execution of TestCases with the VxWorks simulator using an example.

In the first section of this document, testing on a VxWorks target using the animation based testing mode is described. In the second section, testing on a VxWorks target using the assertion based testing mode is described. This new testing mode is available since Rhapsody 7.6.

Execution of TestCases on the VxWorks Target (animation based testing mode)

Please follow the steps as described in the sections below.

Preparing the Code Generation Configuration

- Features dialog of the CG Configuration, tab Settings: Select the correct environment, for example “VxWorks6diab”.
- Properties of the CG Configuration (when testing a C application use the corresponding properties of the subject C_CG instead), please edit the corresponding properties of the environment being used:
 - CPP_CG:VxWorks6diab:RemoteHost
Set the value to the IP of the VxWorks simulator or target. This is needed for the Rhapsody Animation.
 - CPP_CG:VxWorks6diab:DisableDebugBreakpoints
Must be checked.
 - CPP_CG:VxWorks6diab:RTC_ConnectTargetName
Enter the name of the VxWorks simulator. TestConductor automatically connects Rhapsody to the running simulator when starting the TestCase execution if the name is entered here.
 - CPP_CG:VxWorks6diab:RTC_DownloadApplication
Check this property if TestConductor should automatically download the application to the simulator or target.

NOTE: The last three properties are only visible if the configuration is located underneath a TestPackage. If the Configuration is not located underneath a TestPackage, the configuration must be moved inside a TestPackage in order to have these properties available.

Preparing the Test Architecture

- The TestConfiguration of the TestContext shall depend on the CG Configuration with the proper settings for VxWorks. This way the application is started in the VxWorks simulator if the TestCase execution is activated.

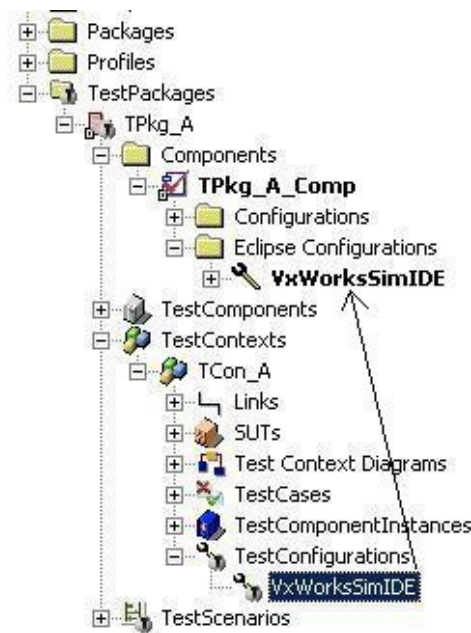


Figure 1: The TestConfiguration depends on the VxWorks CG Configuration

Executing a TestCase

- Start the Wind River WorkBench
- Start the VxWorks simulator
- Connect Rhapsody with the simulator: in Rhapsody, menu Code->Target->Connect (this can be skipped if the property CPP_CG:VxWorks6diab:RTC_ConnectTargetName contains the simulator name)
- Update the TestCase (from the context menu of the TestCase)
- Build the TestCase (from the context menu of the TestCase). it generates code, compiles and builds the application.
- Download the built application to the simulator: in Rhapsody, menu Code->Target->Download (this can be skipped if the property CPP_CG:VxWorks6diab:RTC_DownloadApplication is checked)
- Execute the TestCase (from the context menu of the TestCase): The application is launched on the target (within the VxWorks simulator), while TestConductor is driving and monitoring the TestCase execution on the host machine

Pitfalls and Limitations

Executing TestCases on the Target

- If a TestCase shall be executed TestConductor checks if the executable of the application already exists. If the CG configuration is an Eclipse configuration and the setting “Build configuration in IDE” is active, then TestConductor might search for the executable in the wrong directory and refuses to execute the TestCase. For example TestConductor expects the executable in \$IDEWorkspace/\$IDEProject, while the executable is located in \$IDEWorkspace/\$IDEProject/default. A workaround is to copy the executable to the location \$IDEWorkspace/\$IDEProject.

Execution of TestCases on the VxWorks Target (assertion based testing mode)

Please follow the steps as described in the sections below. Depending on the used environment (VxWorks, VxWorks RTP, VxWorks 653) and compiler (gnu, diab), some properties or tags must be set differently.

This table shows the VxWorks environments supported by TestConductor. For all supported environments, some properties and tags need to be adjusted manually (see details in the next sections).

Environment	C++	C
VxWorks6diab	Supported	Supported
VxWorks6gnu	Supported	Supported
VxWorks6diab_RTP	Supported	Supported
VxWorks6gnu_RTP	Supported	Supported
WorkbenchManaged653	Supported	Supported

File IO needed for assertion based testing

In assertion based testing mode, the tested application must be able to read and write files. Files are used to pass arguments to the tested application and the application writes information about the results of assertions to a file. When computing code coverage, also information used to compute the code coverage is written to a file.

Check the VxWorks documentation for information which folder of the host can be accessed by the target.

If the target being used does not support file IO at all a different setup for assertion based testing has to be applied, using a target proxy as mediator between Rhapsody/TestConductor and the target specific IDE or the target/simulator. See document “Testing with TestConductor on a small target.pdf”.

Environment VxWorks6diab, VxWorks6gnu: Preparing the Code Generation Configuration

Properties of the Code Generation Configuration

Adjust some properties in the CG and in the CPP_CG subject (when testing a C application use the corresponding properties of the subject C_CG instead); example using environment VxWorks6diab (for VxWorks6gnu, use corresponding properties):

- CG::Configuration::StartFrameworkInMainThread
Unchecked
- CG::Configuration::PreFrameworkInitCode
char* argv[9]={}; argv[0]=""; argv[1]=argv1; argv[2]=argv2; argv[3]=argv3;
argv[4]=argv4; argv[5]=argv5; argv[6]=argv6; argv[7]=argv7; argv[8]=argv8;
- CPP_CG::Configuration::MainFunctionArgList
int argc, char* argv1, char* argv2, char* argv3, char* argv4, char* argv5, char* argv6,
char* argv7, char* argv8
- CPP_CG::VxWorks6diab::RemoteHost
Set the value to the IP of the VxWorks simulator or target. This is needed only when using instrumentation mode Animation.
- CPP_CG::VxWorks6diab::RTC_ConnectTargetName
<targetname>@<hostname>
Enter the name of the VxWorks target if TestConductor should automatically connect to the target.
- CPP_CG::VxWorks6diab::RTC_DownloadApplication
Checked, if TestConductor should automatically download the application to the target
- CPP_CG::VxWorks6diab::DisableDebugBreakpoints
Checked
NOTE: The last three properties are only visible if the configuration is located underneath a TestPackage. If the Configuration is not located underneath a TestPackage, the configuration must be moved inside a TestPackage in order to have these properties available.

Tags of the Code Generation Configuration:

- rtc_testexecution_script_content
<WindRiverInstall>\wrenv -p vxworks-6.9 hostShell -c vxmain(9,\"-
resultfile\", \"rtctest.rst\", \"-logfile\", \"rtclog.txt\", \"-tcontext\", \"\$tcontext\", \"-
tcase\", \"\$tcase\") <targetname>@<hostname>
Adjust the variables the following way:
<WindRiverInstall> Replace this with the path to your WindRiver installation
<targetname>@<hostname> Replace this with the name of your VxWorks target

- `rtc_result_filename`
`<HostTempDir>/rtcresult.rst`
Adjust the variables the following way:
`<HostTempDir>` Replace this with the value of the \$TEMP environment variable of your host

After applying these changes, TestConductor tests can be executed on the VxWorks target using the standard TestConductor work flow: Create and specify tests, update, build and execute tests.

Environment VxWorks6diab_RTP, VxWorks6gnu_RTP: Preparing the Code Generation Configuration

Properties of the Code Generation Configuration

Adjust some properties in the CG and in the CPP_CG subject (when testing a C application use the corresponding properties of the subject C_CG instead); example using environment VxWorks6diab_RTP (for VxWorks6gnu_RTP, use corresponding properties):

- CG::Configuration::StartFrameworkInMainThread
Checked (default)
- CPP_CG::Configuration::MainFunctionArgList
int argc, char* argv[] (default)
- CPP_CG::VxWorks6diab_RTP::RemoteHost
Set the value to the IP of the VxWorks simulator or target. This is needed only when using instrumentation mode Animation.
- CPP_CG::VxWorks6diab_RTP::RTC_ConnectTargetName
<targetname>@<hostname>
Enter the name of the VxWorks target if TestConductor should automatically connect to the target.
- CPP_CG::VxWorks6diab_RTP::RTC_DownloadApplication
Checked, if TestConductor should automatically download the application to the target
- CPP_CG::VxWorks6diab_RTP::DisableDebugBreakpoints
Checked
NOTE: The last three properties are only visible if the configuration is located underneath a TestPackage. If the Configuration is not located underneath a TestPackage, the configuration must be moved inside a TestPackage in order to have these properties available.

Tags of the Code Generation Configuration:

- rtc_testexecution_script_content
<WindRiverInstall>\wrenv -p vxworks-6.9 hostShell -m cmd -c
"<CGPath>/<BinaryName>\"-resultfile\" \"rtctest.rst\" \"-logfile\" \"rtclog.txt\" \"-tcontext\" \"\$tcontext\" \"-tcase\" \"\$tcase\"\" <targetname>@<hostname>
Adjust the variables the following way:
<CGPath> Replace this with the path to the CG folder on your host. Use forward slashes for this path (like "C:/model").
<BinaryName> Replace this with the file name of the application binary
<WindRiverInstall> Replace this with the path to your WindRiver installation
<targetname>@<hostname> Replace this with the name of your VxWorks target

- `rtc_result_filename`
`<HostTempDir>/rtcresult.rst`
Adjust the variables the following way:
`<HostTempDir>` Replace this with the value of the \$TEMP environment variable of your host

After applying these changes, TestConductor tests can be executed on the VxWorks target using the standard TestConductor work flow: Create and specify tests, update, build and execute tests.

Environment VxWorks6diab, VxWorks6diab_RTP, VxWorks6gnu, VxWorks6gnu_RTP: Executing a TestCase

- Start the Wind River WorkBench
- Start the VxWorks simulator/target if necessary
- Connect Rhapsody with the simulator: In Rhapsody, menu Code->Target->Connect (not needed if property CPP_CG::VxWorks6diab::RTC_ConnectTargetName contains the correct name of the target)
- Update the TestCase (from the context menu of the TestCase)
- Build the TestCase (from the context menu of the TestCase). It generates code, compiles and builds the application.
- Download the built application to the simulator: In Rhapsody, menu Code->Target->Download (not needed if property CPP_CG::VxWorks6gnu_RTP::RTC_DownloadApplication is checked)
- Execute the TestCase (from the context menu of the TestCase). The application is launched on the target, while TestConductor is driving and monitoring the TestCase execution on the host machine. TestConductor shows the status of the test execution in the test execution window.
- Inspect the result of the TestCase execution: TestConductor automatically adds the detailed html result for the TestCase execution to the Rhapsody model.

Also, execution of several TestCases in a row is possible by invoking “Execute TestContext” on a TestContext or by invoking “Execute TestPackage” on a TestPackage.

Computation of code coverage

Computation of code coverage using the gnu compiler is supported for the Rhapsody environments VxWorks6gnu and VxWorks6gnu_RTP for C++ and C.

Computation of code coverage for the diab compiler is supported for the Rhapsody environments VxWorks6diab and VxWorks6diab_RTP for C++ and C.

Target configuration

For computation of code coverage, the source code needs to be instrumented (annotated) with some macros which are used to track the executed functions, statements, decision branches. For a correct annotation TestConductor needs some information about the compiler and the target system, the so called target configuration: The size and sign of some types, the endian of the target, the compiler family and some more. To collect this information a small program must be compiled (with the same compiler and compiler options which are used to compile the tested application) and executed on the same target the tested application will be executed on. The target configuration tool will collect the needed information and write it into an xml file, this file can be used from then on for the instrumentation for code coverage until the target configuration (compiler, compiler options, target) changes.

To build and execute the target configuration tool a Rhapsody model can be used which is part of the TestConductor installation, in folder TestConductor/CodeCoverage/TargetConfiguration. Copy the folder TargetConfiguration to a folder which can be written to and open the project in Rhapsody in C++ or C. The project contains one Code Generation Component for C++ and one for C, each with several configurations for different environments. Set the Code Generation Configuration predefined for the environment to be used as active configuration and adjust the settings in the properties according to your environment: Compiler options, information about the VxWorks installation (like BSP property). Generate code and build the tool.

Now load the tool on the target and execute it. When using VxWorks6diab_RTP or VxWorks6gnu_RTP the name of the output xml file has to be provided as command line argument. When using VxWorks6diab or VxWorks6gnu the file name is per default targetconf.xml, this can be modified by editing the targetconf.cpp or targetconf.c source code file. Check VxWorks documentation for the default path for files created by applications running on the target or how to modify this path.

The target configuration tool will collect the necessary data and write it to the output xml file. Copy this file to a location which can be accessed during compilation of the application you want to test, for example in the main or code generation folder of the application's project.

Options file for computation of code coverage (gnu compiler)

To compute code coverage, the user must provide some information about the installation of the VxWorks environment and the VxWorks version in an xml options file.

A template for an options file with some comments is provided in the TestConductor installation: Copy file **<RhapsodyInstall>/TestConductor/TCCodeAnnotationOptions.xml** to another location (for example, into the main folder of the Rhapsody project). Open the copy in an editor and enter the needed attributes and values in the **<Environment>** section:

- Attribute **<Compiler>**
 - `name="GNU"`
 - `cppcompiler=` When using C++, enter the name of the VxWorks gnu C++ compiler (example: `c++pentium.exe`).
 - `ccompiler=` When using C, enter the name of the VxWorks gnu C compiler (example: `ccpentium.exe`).
- Attribute **<TargetConfigFile>**
 - `relative_path=` Enter the path and file name of the target configuration xml file, relative to the code generation folder.

Alternatively:
 - `absolute_path=` Enter the full path of the target configuration xml file.
- Attribute **<HostToolsEnvironment>**
 - `name="VxWorks"`
- Attribute **<VxWorks>**
 - `wrenv_path=` Enter the full path of the Wind River environment tool installed on the host (example: `C:\WindRiver\wrenv.exe`).
 - `package=` Enter the VxWorks package to be used (example: `vxworks-6.9`).
 - `host_tmp_dir=` Enter the path to a folder on the host which can be used to exchange files between the host and the target/simulator (example: The value of the \$TMP environment variable, like `C:\Users\John\AppData\Local\Temp`).

See figure 2 below for an example of an options file for computation of code coverage using the gnu compiler.

```

<?xml version="1.0" encoding="UTF-8"?>
<CodeGeneration xmlns:cq="http://www.btc-es.de/CodeGeneration/1.0">
  <!-- Settings for compile environment and host system. -->
  <Environment>
    <!-- Allowed Compiler name: MSVC, GNU, DIAB, INTEGRITY or ANSI. -->
    <!-- For MSVC, cppcompiler is the command name of the C++ compiler (default: cl.exe) and ccompiler is the command name of the C compiler. -->
    <!-- For GNU, cppcompiler is the command name of the C++ compiler and ccompiler is the command name of the C compiler. -->
    <!-- For DIAB, cppcompiler is the command name of the C++ compiler and ccompiler is the command name of the C compiler. -->
    <!-- For INTEGRITY, cppcompiler is the command name of the C++ compiler and ccompiler is the command name of the C compiler. -->
    <Compiler name="GNU" cppcompiler="c++pentium.exe"/>
    <!-- Specify a file containing type information for the target. -->
    <!-- absolute_path: The full path name of the file. -->
    <!-- relative_path: Path of file relative to main folder of CG Configuration. -->
    <TargetConfigFile relative_path="../../targetconf_gnu_cpp.xml"/>
    <!-- Allowed HostToolsEnvironment name: Cygwin, VxWorks or GHS. -->
    <HostToolsEnvironment name="VxWorks"/>
    <!-- When compiling for VxWorks (VxWorks general or VxWorks 653) : -->
    <!-- wrenv_path: Specify the full path to the wrenv.exe tool (example: C:\WindRiver\wrenv.exe) -->
    <!-- package: Specify the name of the used vxworks package (example: vxworks-6.9). -->
    <!-- host_tmp_dir: Specify the path of the TMP dir of the host (will be used to exchange files) -->
    <VxWorks wrenv_path="C:\WindRiver69\wrenv.exe" package="vxworks-6.9" host_tmp_dir="C:/tmp"/>
  </Environment>
</CodeGeneration>

```

Figure 2: Code coverage options to provide information about the VxWorks environment when using GNU compiler (example for C++).

In the Rhapsody model, use a tag of the Code Generation configuration to specify the path to the options file: Open the feature dialog of the Code Generation configuration and go to the Tags section. Then enter the path (including name and extension) to the options file into the tag CodeCoverageOptionsFilename. You can use an absolute path or a path relative to the code generation main folder (location of the Makefile).

Options file for computation of code coverage (diab compiler)

To compute code coverage, the user must provide some information about the installation of the VxWorks environment and the VxWorks version in an xml options file.

A template for an options file with some comments is provided in the TestConductor installation: Copy file <RhapsodyInstall>/TestConductor/TCCodeAnnotationOptions.xml to another location (for example, into the main folder of the Rhapsody project). Open the copy in an editor and enter the needed attributes and values in the <Environment> section:

- Attribute <Compiler>
 - name="DIAB"
 - cppcompiler= When using C++, enter the name of the VxWorks diab C++ compiler (example: dplusplus.exe).
 - ccompiler= When using C, enter the name of the VxWorks diab C compiler (example: dcc.exe).
- Attribute <TargetConfigFile>

- `relative_path`= Enter the path and file name of the target configuration xml file, relative to the code generation folder.

Alternatively:

- `absolute_path`= Enter the full path of the target configuration xml file.
- Attribute `<HostToolsEnvironment>`
 - `name`="VxWorks"
- Attribute `<VxWorks>`
 - `wrenv_path`= Enter the full path of the Wind River environment tool installed on the host (example: C:\WindRiver\wrenv.exe).
 - `package`= Enter the VxWorks package to be used (example: vxworks-6.9).
 - `host_tmp_dir`= Enter the path to the folder on the host which is used to exchange files between the host and the target/simulator (example: The value of the \$TMP environment variable, like C:\Users\John\AppData\Local\Temp).

See figure 3 below for an example of an options file for computation of code coverage.

```
<?xml version="1.0" encoding="UTF-8"?>
<CodeGeneration xmlns:cg="http://www.btc-es.de/CodeGeneration/1.0">
  <!-- Settings for compile environment and host system. -->
  <Environment>
    <!-- Allowed Compiler name: MSVC, GNU, DIAB, INTEGRITY or ANSI. -->
    <!-- For MSVC, cppcompiler is the command name of the C++ compiler (default: cl.exe) and ccon
    <!-- For GNU, cppcompiler is the command name of the C++ compiler and ccompiler is the commar
    <!-- For DIAB, cppcompiler is the command name of the C++ compiler and ccompiler is the comm
    <!-- For INTEGRITY, cppcompiler is the command name of the C++ compiler and ccompiler is the
    <Compiler name="DIAB" cppcompiler="dplus.exe"/>
    <!-- Specify a file containing type information for the target. -->
    <!-- absolute_path: The full path name of the file. -->
    <!-- relative_path: Path of file relative to main folder of CG Configuration. -->
    <TargetConfigFile relative_path="../../targetconf_diab_cpp.xml"/>
    <!-- Allowed HostToolsEnvironment name: Cygwin, VxWorks or GHS. -->
    <HostToolsEnvironment name="VxWorks"/>
    <!-- When compiling for VxWorks (VxWorks general or VxWorks 653) : -->
    <!-- wrenv_path: Specify the full path to the wrenv.exe tool (example: C:\WindRiver\wrenv.exe
    <!-- package: Specify the name of the used vxworks package (example: vxworks-6.9). -->
    <!-- host_tmp_dir: Specify the path of the TMP dir of the host (will be used to exchange file
    <VxWorks wrenv_path="C:\WindRiver69\wrenv.exe" package="vxworks-6.9" host_tmp_dir="C:/tmp"/>
  </Environment>
</CodeGeneration>
```

Figure 3: Code coverage options to provide information about the VxWorks environment when using DIAB compiler.

In the Rhapsody model, use a tag of the Code Generation configuration to specify the path to the options file: Open the feature dialog of the Code Generation configuration and go to the Tags section. Then enter the path (including name and extension) to the options file into the tag `CodeCoverageOptionsFilename`. You can use an absolute path or a path relative to the code generation main folder (location of the Makefile).

Building and executing tests with computation of code coverage

After providing these information, update, build and execute the TestConductor tests to compute code coverage information. After the execution of the tests has finished, TestConductor automatically adds a detailed code coverage report to the Rhapsody model.

Environment WorkbenchManaged653: Preparing the Code Generation Configuration

Settings of the Code Generation Configuration:

- Standard Headers: <stdio.h>,<string.h>

Properties of the Code Generation Configuration

Adjust some properties in the CG and in the CPP_CG subject (when testing a C application use the corresponding properties of the subject C_CG instead):

- CG::Configuration::StartFrameworkInMainThread
Unchecked
- CG::Configuration::PreFrameworkInitCode

```
char c1[200];
char c2[200];
FILE *f;
int argc;
char* argv[]={ "", "-resultfile", "/tgtsvr/TestApplication_Partition/rtrresult.rst", "-logfile", "/tgtsvr/TestApplication_Partition/rtrlog.txt", "-tcontext", c1, "-tcase", c2 };
f = fopen("/tgtsvr/TestApplication_Partition/test_args.txt", "r");
if (f != NULL) {
    if(fgets(c1, 200, f)) {
        int i = strlen(c1);
        if (c1[i - 1] == '\n')
            c1[i - 1] = '\0';
        if (c1[i - 2] == '\r')
            c1[i - 2] = '\0';
    }
    if(fgets(c2, 200, f)) {
        int i = strlen(c2);
        if (c2[i - 1] == '\n')
            c2[i - 1] = '\0';
        if (c2[i - 2] == '\r')
            c2[i - 2] = '\0';
    }
    fclose(f);
}
argc=9;
```

TestApplication_Partition Folder containing the code of the application partition of the tested project.

This code is necessary to pass arguments to the tested application using a file (passing arguments to the application on the command line is not supported). The file is read by the application via the WindRiver target server.

While the test is executed on the target or simulator, the files rtcresult.rst and rtclog.txt with data about the test execution will be written to the Workspace folder on the host via the WindRiver target server.

Please check for erroneous new lines after copying above code from this document into the property.

- **CPP_CG::WorkbenchManaged653::DisableDebugBreakpoints**
Checked

Note: This property is only visible if the configuration is located underneath a TestPackage. If the Configuration is not located underneath a TestPackage, the configuration must be moved inside a TestPackage in order to have these properties available.

Tags of the Code Generation Configuration:

- **rtc_testexecution_script_content**

```
C:
cd WorkspaceDir/TestApplication_Partition
echo $context>test_args.txt
echo $case>>test_args.txt
echo start vxSim -f WorkspaceDir/IntegrationProject/boot.txt > run_tests.bat
echo tgtsvr.exe -n VxSim0 -B wdbpipe -V -R WorkspaceDir -RW -Bt 3 -A VxSim0 >> run_tests.bat
WrenvInstallDir\wrenv.exe -p vxpackage WorkspaceDir/TestApplication_Partition/run_tests.bat
```

WorkspaceDir Replace this with the path of the workspace folder on your host, like C:\WindRiver\workspace

TestApplication_Partition Folder containing the code of the application partition of the tested project.

IntegrationProject Folder containing the main files of the integration project.

WrenvInstallDir The installation folder of the Wind River environment tool.

vxpackage Name of the VxWorks 653 package to be used, like Vxworks653-2.3.

Note: If the workspace is not located on drive C, please adjust also the first line according to the drive the workspace is located on.

- **rtc_result_filename**
WorkspaceDir/TestApplication_Partition/rtcresult.rst

WorkspaceDir Replace this with the path of the workspace folder on your host, like C:\WindRiver\workspace

TestApplication_Partition Folder containing the code of the application partition of the tested project.

- **rtc_exit_kind**
user_defined
- **rtc_exit_user_definition**
RETURN_CODE_TYPE retCode = NO_ERROR;\nSET_PARTITION_MODE (NORMAL, &retCode);

Make sure the tags, settings and other options for IDE integration are set correctly to be able to generate code from the code generation configuration into the workspace folder for the application partition.

After applying these changes, execution of TestConductor tests for VxWorks 653 target can be prepared using this workflow:

- Create and specify tests in Rhapsody
- Update tests in Rhapsody
- Build tests in Rhapsody: This will generate the code of the tested application into the workspace folder of the application partition. Building the code must be performed in the WorkBench IDE.

Environment WorkbenchManaged653: Building and executing a TestCase

- Start the Wind River WorkBench.
- In the WorkBench IDE, refresh the project to update the project data with the code generated from Rhapsody.
- To be able to execute TestConductor tests, the application must support File I/O because during execution of tests, status information for the result evaluation is written into a file. If the VxWorks 653 application does not already support File I/O, please do this in the Workbench IDE:
 - In the Project Explorer, go to the Module OS partition, expand this node.
 - Right click Kernel Configuration, select Edit Kernel Configuration.
 - In the Kernel Configuration editor, expand node development tool components.
 - Right click Windview components, select Include.
 - In the opening dialog, the default selection is sufficient: Click Finish.
 - In the Project Explorer, right click ADD_NEEDED, select Build target.
- In the WorkBench IDE, build the project.
- In Rhapsody, execute the TestCase (from the context menu of the TestCase). A command window running a batch file opens. TestConductor waits for the tested application to finish the TestCase as long as this window is open.
- The simulator with the tested application is started from the batch file.
- The target server is started from the batch file.
- When the TestCase has finished, a message is printed to the console of the target/simulator: Finished test case <number>. Now close the simulator to notify TestConductor the execution of the TestCase has finished.
- TestConductor evaluates the file with the data from the test execution and create the html result. The status of the test execution is shown in the test execution window.
- Inspect the result of the TestCase execution: TestConductor automatically adds the detailed html result for the TestCase execution to the Rhapsody model.

Environment WorkbenchManaged653: Computation of code coverage

Computation of code coverage is supported for the Rhapsody environments WorkbenchManaged653 for C++ and C.

Target configuration

Prepare the file containing the target configuration as described in Target configuration.

One additional step is necessary before building the target configuration tool when using VxWorks 653: The output path for the xml file must be entered manually. Open the source code in an editor, search for the string “EDIT_PARTITION_FOLDER” and replace it by the path of the application partition.

Options file for computation of code coverage

To compute code coverage, the user must provide some information about the installation of the VxWorks 653 environment and the VxWorks 653 version in an xml options file.

A template for an options file with some comments is provided in the TestConductor installation: Copy file <RhapsodyInstall>/TestConductor/TCCodeAnnotationOptions.xml to another location (for example, into the main folder of the Rhapsody project). Open the copy in an editor and enter the needed attributes and values in the <Environment> section:

- Attribute <Compiler>
 - name=”GNU”
 - cppcompiler= When using C++, enter the name of the VxWorks 653 gnu C++ compiler (example: ccpentium.exe).
 - ccompiler= When using C, enter the name of the VxWorks 653 gnu C compiler (example: ccpentium.exe).
- Attribute <TargetConfigFile>
 - relative_path= Enter the path and file name of the target configuration xml file, relative to the code generation folder.

Alternatively:

- absolute_path= Enter the full path of the target configuration xml file.
- Attribute <HostToolsEnvironment>
 - name=”VxWorks”

- Attribute <VxWorks>
 - wrenv_path= Enter the full path of the Wind River environment tool installed on the host (example: C:\WindRiver\wrenv.exe).
 - package= Enter the VxWorks package to be used (example: vxworks-6.9).
 - host_tmp_dir= Enter the path to the folder on the host which is used to exchange files between the host and the target/simulator (example: The value of the \$TMP environment variable, like C:\Users\John\AppData\Local\Temp).
- Attribute <VxWorks653>

To be able to compute code coverage, some data needs to be transferred from the tested application on the target/simulator to TestConductor on the host using a file. The VxWorks 653 application can write the file via the Wind River target server. The folder for this file must be entered in two attributes, once in standard notation of the host operating system (host path) and once in target server notation (target path):

 - host_data_dir= Enter the path to the folder which is used to exchange files between the tested application and the host, in host notation (example: C:/wr-workspace/MyProject_Partition1).
 - target_data_dir= Enter the path to the folder which is used to exchange files between the tested application and the host, in target server notation (example: /tgtsvr/MyProject_Partition1).

See figure 4 below for an example of an options file for computation of code coverage.

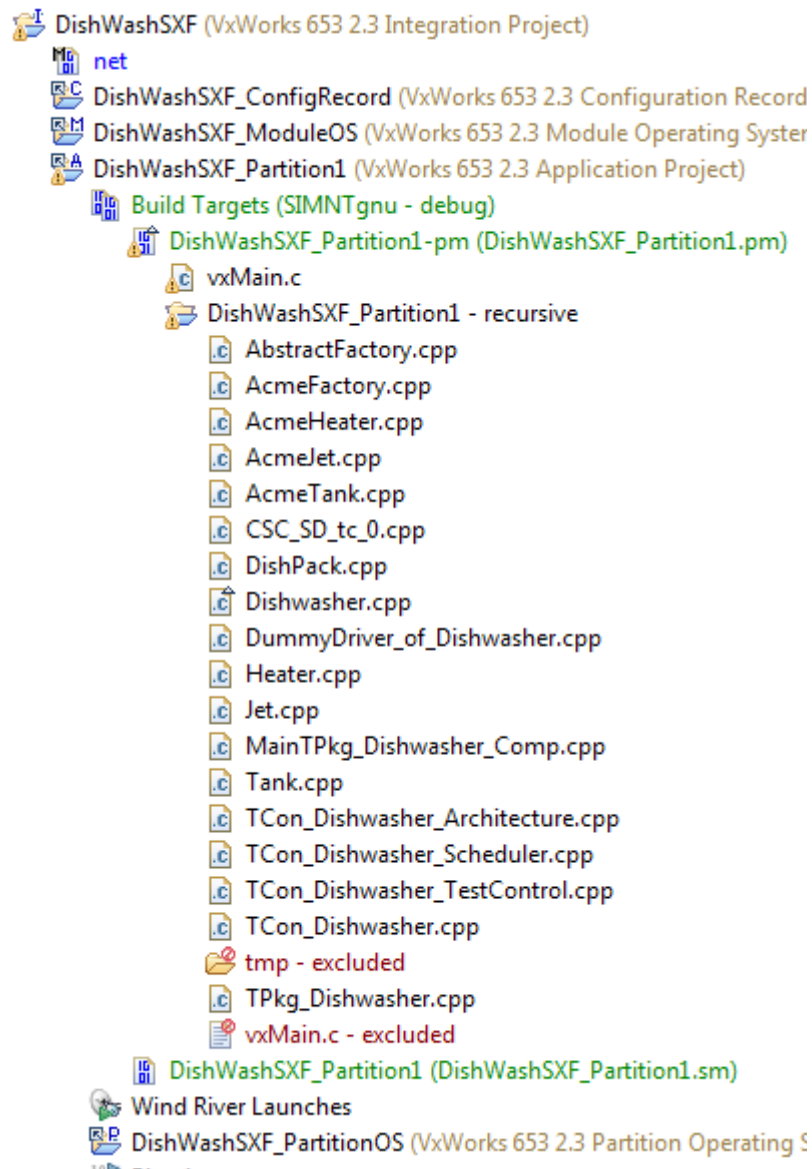
```
<?xml version="1.0" encoding="UTF-8"?>
<CodeGeneration xmlns:cg="http://www.btc-es.de/CodeGeneration/1.0">
  <!-- Settings for compile environment and host system. -->
  <Environment>
    <Compiler name="GNU" cppcompiler="ccpentium.exe" />
    <!-- Specify a file containing type information for the target. -->
    <!-- absolute_path: The full path name of the file. -->
    <!-- relative_path: Path of file relative to main folder of CG Configuration. -->
    <TargetConfigFile relative_path="./targetconf_vxworks653_cpp.xml"/>
    <HostToolsEnvironment name="VxWorks"/>
    <!-- When compiling for VxWorks (VxWorks general or VxWorks 653) : -->
    <!-- wrenv_path: Specify the full path to the wrenv.exe tool (example: C:\WindRiver\wrenv.exe). -->
    <!-- package: Specify the name of the used vxworks package (example: vxworks-6.9). -->
    <!-- host_tmp_dir: Specify the path of the TMP dir of the host (will be used to exchange files between target and host). -->
    <VxWorks wrenv_path="C:\WindRiver653\wrenv.exe" package="vxworks653-2.3"/>
    <!-- Additional options for VxWorks 653 environment: -->
    <!-- host_data_dir: Path to a directory which can be used to exchange files between target and host (example: C:/workspace/project_
    <!-- target_data_dir: The same directory in target server notation (example: /tgtsvr/project_partition1). -->
    <VxWorks653 host_data_dir="C:/Workspace-653/DishWasherDemoSXF_Partition1" target_data_dir="/tgtsvr/DishWasherDemoSXF_Partition1"/>
  </Environment>
</CodeGeneration>
```

Figure 4: Code coverage options to provide information about the VxWorks 653 environment when using GNU compiler.

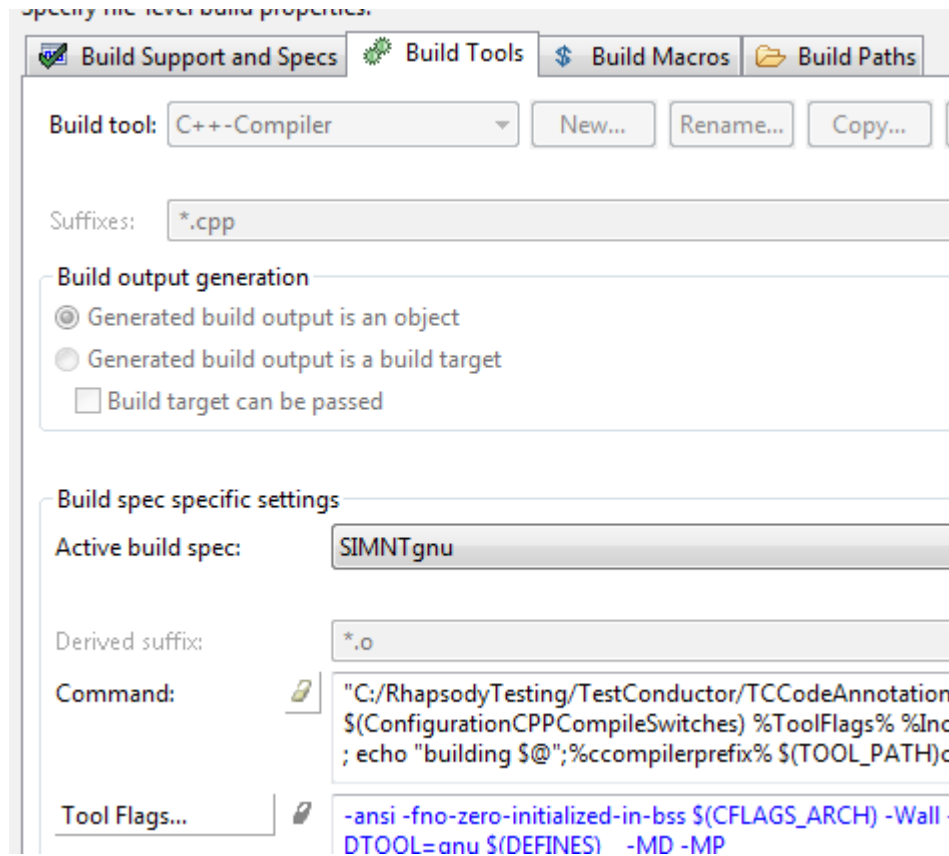
In the Rhapsody model, use a tag of the Code Generation configuration to specify the path to the options file: Open the feature dialog of the Code Generation configuration and go to the Tags section. Then enter the path (including name and extension) to the options file into the tag CodeCoverageOptionsFilename. You can use an absolute path or a path relative to the code generation main folder (location of the Makefile).

Modify compile command in Workbench project

To be able to compute code coverage, the compile command for the system under test (SUT) must be modified in the Wind River Workbench project. In the Workbench, expand the Application Project, then expand the Build Targets and the nested elements until you see the list of source code files of the application (see example in figure 5).



Now right click the source code file of the SUT and select menu Properties. In the opened dialog, select tab Build Tools and modify the compile command in the edit field right to Command: (see figure 6).



Before the original compile command in this edit field, the following text must be entered (with the path to the Rhapsody installation adjusted):

```
"<RhapsodyInstall>/TestConductor/TCCodeAnnotation.exe" "$(PRJ_ROOT_DIR)" $(ConfigurationCPPCompileSwitches) %ToolFlags% %Includes% $(ADDED_INCLUDES) %InFile% ;
```

After the original compile command in this edit field, the following text must be entered:

```
; if [ -f "%InFile%.bak" ]; then cp "%InFile%" "%InFile%.annotated"; fi ; if [ -f "%InFile%.bak" ]; then mv "%InFile%.bak" "%InFile%"; fi
```

These commands will be generated into the Makefiles by the Workbench. Before the SUT code is compiled it is instrumented (annotated) with additional code responsible for collecting the coverage information, this is done by the TestConductor tool TCCodeAnnotation.exe. Then the annotated code is compiled (using the original command) and after this a backup file of the original code is copied back to the original file name.

After providing these information, update, build and execute the TestConductor tests to compute code coverage information. After the execution of the tests has finished, TestConductor automatically adds a detailed code coverage report to the Rhapsody model.

Avoid compile errors due to annotated code in tmp folder

When annotating the source code to prepare computation of code coverage, TestConductor writes some files with the extension “.c” into a folder tmp. Depending on the settings in the coverage options file, this tmp folder might be located in the folder containing the source code of the tested application (see figure 5, the red tmp folder). In this case, the Workbench IDE automatically adds any *.c file in this folder to the Makefile when building this application which can cause compiler or linker errors.

To avoid this, this tmp folder should be excluded from the build. To do this, right click the tmp folder and select Exclude from Build Target.

Limitations

Support for VxWorks 653 (Rhapsody environment WorkbenchManaged653)

- TestConductor does not support VxWorks 653 platform in animation based testing mode.
- TestConductor does not support full integration of testing for VxWorks 653 platform in assertion based testing mode: Some settings must be done manually (in the Rhapsody project or in the Workbench IDE).