BIO

PAPER

F5

November 8, 2002
10:15 AM

TESTING A DISTRIBUTED APPLICATION WRITTEN IN EMBEDDED WINDOWS CE

Jim Walters
BSQUARE Corporation

International Conference On
Software Testing Analysis & Review
November 4-8, 2002
Anaheim, CA USA

Jim Walters

Jim Walters directs the SmartBuild Software Development team in BSQUARE's Platform Systems Division. Mr. Walters has spent more than 18 years in the technology industry with experience ranging from hands-on black box testing, to the design and implementation of proprietary and commercial test automation systems, to serving as business director for a multi-million developer tool product line. Walters has worked for several large software houses including Microsoft, Aldus, Adobe, and Apple Computer. Mr. Walters holds a Bachelor of Science in Mathematics from The Evergreen State College.

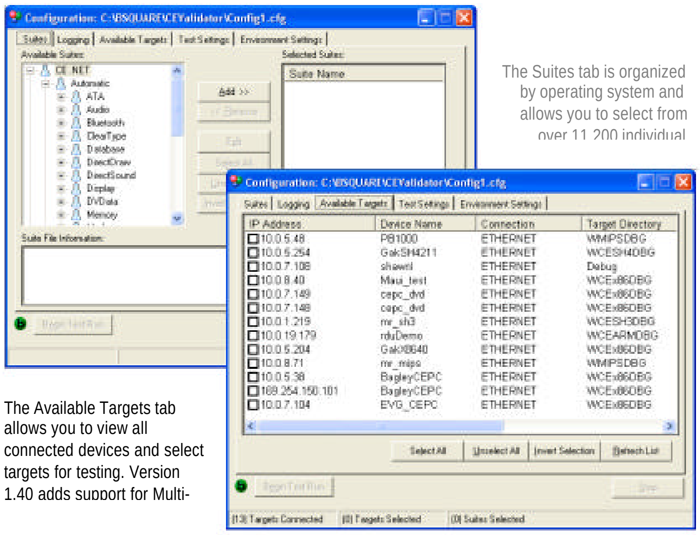
Testing Microsoft® Windows® CE-based Smart Devices

By Jim Walters, Director, SmartBuild Software Development

For many of today's smart devices, Windows CE is the new operating system of choice. Increasingly, developers are embedding Windows CE into many different types of appliances, including set-top boxes, gaming systems, bar code scanners, factory automation systems, and many more. As Windows CE has grown, so too has the need for off-the-shelf software development tools. Today there are many tools and off-the-shelf software kits on the market that save device designers significant amounts of time and effort throughout the device development process.

Now, as more and more of these devices reach the final stages of development a new significant need has arisen: the need for testing tools. Until recently, there were only two testing options available to developers of Windows CE-based devices: write test code in-house or contract a firm to develop test code on a custom basis. To complete the testing project in-house, Original Equipment Manufacturers (OEMs) must train their staff, spend time developing testing software, and then after months of preparation, begin testing their devices. Or, they must contract with a software engineering firm to spend months developing custom testing software. Either way, testing Windows CE-based devices has become a costly, time-consuming process.

Now organizations can take advantage of a more cost- and time-effective solution for testing Windows CE-based smart devices; CEV (CEV) PassPORT. CEV PassPORT enables faster, and more accurate testing of a port of Windows CE to target hardware. The result of more than 10 person years of development efforts, CEV PassPORT is has been thoroughly tested on every supported CPU platform and dozens of machine implementations.



IP Address	Device Name	Connection	Target Directory
10.0.5.48	PB1000	ETHERNET	WMIPSDBG
10.0.5.254	GakSH4211	ETHERNET	WCESH4DBG
10.0.7.108	shawnl	ETHERNET	Debug
10.0.8.40	Maui_test	ETHERNET	WCEx86DBG
10.0.7.149	cepc_dvd	ETHERNET	WCEx86DBG
10.0.7.148	cepc_dvd	ETHERNET	WCEx86DBG
10.0.1.219	mr_sh3	ETHERNET	WCESH3DBG
10.0.19.179	rduDemo	ETHERNET	WCEARMDBG
10.0.5.204	GakX8640	ETHERNET	WCEx86DBG
10.0.8.71	mr_mips	ETHERNET	WMIPSDBG
10.0.5.38	BagleyCEPC	ETHERNET	WCEx86DBG
169.254.150.101	BagleyCEPC	ETHERNET	WCEx86DBG
10.0.7.104	EVG_CEPC	ETHERNET	WCEx86DBG

[13] Targets Connected   [0] Targets Selected   [0] Suites Selected

The Suites tab is organized by operating system and allows you to select from over 11,200 individual

The Available Targets tab allows you to view all connected devices and select targets for testing. Version 1.40 adds support for Multi-

Developed From Experience

BSQUARE completes dozens of Windows CE integration projects each year and recognized early on that an automated test solution would soon be needed for testing Windows CE-based devices. Extensively borrowing “best of breed” methodologies from the experiences of team members as well as the proven automation techniques from BSQUARE’s Tools division, an internal tool was developed which cut the time to test a Windows CE adaptation in half. That internal tool was carefully tested and transformed into the world’s first automated testing product for Windows CE projects: CEV PassPORT.

Over 11,200 Tests Provide Comprehensive Validation

To properly test the port of Windows CE, you will need literally hundreds of tests. CEV PassPORT is a collection of over 11,2000 tests, grouped by the OS subsystem they verify. The product includes both test sources and executables for all tests. The collection covers the major Windows CE subsystems and the common adaptation drivers, with a special emphasis on subsystems, which historically have exhibited the most problems. Subsystems tested by CEV PassPORT include: NDIS, serial port driver, display driver, touch panel driver, mouse driver, keyboard driver, OEM Adaptation Layer code, and PC Card adapter driver.

Test Early and Often To Smooth The Process

In QA it is best to test early and often to identify small problems before they become huge behemoths. CEV PassPORT not only enhances the quality of Windows CE-based smart devices product but it shortens the QA process by enabling frequent and painless testing often throughout the QA process, catching problems early.

The Tests

The CEV PassPORT tests are designed to run "standalone." Standalone means they can be executed outside of the CEValidator infrastructure. For test development it is convenient to develop and test new tests without requiring knowledge of an elaborate infrastructure. Perhaps more important than easing the development of new tests, running standalone makes debugging of reported defects considerably simpler than it would be if one had to wade through and factor out test system infrastructure code.

The sole implementation requirement for a test executable is that test case passes and failures be reported using two specific APIs: WRITETESTPASS() and WRITETESTFAIL(). These macros have signatures similar to the well-known printf() function but their use: 1) generates a standard test case result format in the results file amenable to automated summarization and 2) integrate the reporting with the host user interface.

The CEV PassPORT suites mediate between the graphical user interface and the test cases encapsulated in the test executables. Or, more accurately, they are the means by which the user interface distributes and coordinates the execution of the tests. The suites are text files composed of suite language commands and the commands are the direct expressions of the tasks needed to distribute and execute tests. A test needs to be put on the target device. Then the executable needs to be run. Wait for the test to quit running. And finally, delete files from the device’s file system to restore the file system to its initial state. PUT, RUN, WAIT and DELETE are the basic suite language commands.

Other supported suite commands are:

Command	Description	Notes
GET	Retrieves a file from the device.	
RUNHOST	Run an executable on the host computer.	Useful for client/server style tests.
WAITHOST	Wait for termination of process on the host computer.	Useful for client/server style tests.
PUTSYSTEM	Put a file in the device's system directory.	(\Windows)
SLEEP	Your basic timing function for when all else fails.	
MSGBOX	Displays a message box on the host machine.	Useful for coordinating user actions.
SETREG	Adds or changes a registry setting on a device.	
DELREG	Removes a registry setting on the device.	

Initial suite file comments, in addition to providing internal suite documentation, are presented as the suite descriptions in the user interface.

A sample suite is provided below:

```
'Sample Program to demonstrate the logging format and features.
'Intended to help new users understand and develop log files.
'
' Refer to Help Reference Document for more information

'SETREG HKEY_LOCAL_MACHINE\Debug
"JITDebugger"="WCEJITDebug.exe"
'PUTSYSTEM wcejitdebug.exe

PUT SampleTest.exe
RUN SampleTest.exe
WAIT SampleTest.exe

DELETE SampleTest.exe

'Done with sample test script
```

Reflecting the CEV PassPORT design principle of providing maximum flexibility while still affording great ease-of-use by presenting an intuitive interface to the user, the suite files are organized by their hierarchical placement within a directory structure on the host. The directory and suite file names are the descriptive elements in the user interface. To change the organization of the suites and the user interface titles of the suite files merely requires changing the structure of the directory containing the suite files and the suite file filenames. The user interface automatically reflects changes in the suite file organization.

By default, CEV PassPORT's suites are divided at the top level as being either automatic, manual or stress suites. Automatic suites are those tests that do not require any user intervention during their execution. Manual tests, on the other hand, require some sort of user interaction. Examples of suites that run manual tests are some of the keyboard driver suites and the touch panel suites.

The third top-level category of suites in CEV PassPORT contains the "stress" suites. These suites batter the system by pressing I/O throughput to the limit, operating with little to no available program or object store memory and can bring a custom file system to its knees with multithreaded concurrent abuse. The suites can take a substantial amount of time to run and have been put in their own “not for the faint-hearted” category.

Beneath the top-level organization of automatic, manual and stress the suites are arranged by tested subsystem. For most subsystems there exists further refinements of the organization. The organization makes it easy to locate a functional areas specific suite.

CEV PassPORT’s suites and tests have been run on dozens of different Windows CE platforms. They are the tests used on many of BSQUARE’s products and on our OEM adaptations. Continuously reviewed and extended, the latest and greatest baseline of CE tests are available to our CEV PassPORT customers.

The Graphical User Interface

Of course, all the automated test executables in the world will not help if they are never executed. A simple-to-use framework driven by an intuitive user interface facilitates the frequent and timely use of CEV PassPORT.

The user interface for the CEV PassPORT host component is a standard Windows application. It leverages the well-known Microsoft Windows user interface and makes the execution of suites and collecting of logging results so simple even my manager can do it.

The multiple document interface conveniently represents configurations. Although the user interface can be mastered in just a couple of hours, the user interface includes extensive online help, a bazillion tool tips and an optional wizard that steps the user through the creation of configurations. Asynchronously updated status information keeps tabs on available target devices, suite execution progress and displays the results the color-coded test results as they are generated.

In the terminology of CEV PassPORT a configuration consists of a collection of suites to run, a list of target devices to test and various execution options, which define such things as the desired location for the result log files and stress conditions for the test run. Configurations can be saved and reused during subsequent CEV PassPORT sessions. Configuration parameters are logically organized within a configuration’s tabbed window.

In the configuration window’s Suite tab a hierarchical view of the available suites is presented as well as an editable list of the suites to be run. Adding suites to the run list is as easy as clicking on a suite or a branch representing a functional area and clicking the Add button. Particularly randy users have been know to select all Automatic tests in a simple pop. Masochists tend to favor the top-level Stress group. For the user’s convenience specific information about selected suites and the functional area branches of the suite tree is displayed dynamically as the items are selected.

CEV PassPORT distributes suites as client/server application. The graphical user interface being described here interacts with a small application running on the target device (CEHarness.exe). Since this communication can occur over Ethernet it is possible to have one or more hosts running suites against one or more available target devices. In particular, if several target devices are available either multiple configurations can be run concurrently against different targets or, perhaps of greater utility, suites can be doled out by a configuration on a first-come-first-served basis to several devices. When more than one device is available for testing this reduces the cycle time for running a set of suites. Another useful aspect of using TCP/IP for communication between the host and the target device is that the two device computers do not need to be co-located. As is often the case with new devices, prototypes are far and few between. TCP/IP over Ethernet makes sharing a prototype machine easy.

Other tabs in the configuration window contain various suite execution and results logging options. For example, it can be specified whether result logs are collected on the host or are left on the device and which directory the log files should be stored in. Another tab in the configuration allows the user to set stress conditions for the configuration run. The CPU can be loaded with several high priority threads during the suite executions. Another option reduces the available program memory and storage memory. And yet another option causes the set of suites to be run repeatedly until the user explicitly clicks on the always-available "Stop" button.

"Infinite loop" testing is a good way to find memory or other system resource leaks in the drivers. Tests without clear and useful reported results are nearly worthless. Much work went into having the CEV PassPORT tests generate useful error information when tests fail. While suites are running tests results are displayed both in dynamically created log windows and in a configuration's summary tab. The logging windows contain the full text of a test's results. Failures are color coded red for easy identification. Navigation buttons in the logging window allow you to quickly move from failure to failure. The logging APIs in the tests also cause a prolog and epilog to be generated in each result file. Information such as concurrently running processes, battery power level and date and time of execution is automatically recorded in the results file and displayed in the log window. Useful summary information such as loss of program or storage memory and the total time for test execution is provided in a tab on the log window.

The summary information for a test result is also collected and displayed in a Summary tab of the configuration window. The summary tab reports in real time the number of PASS and FAIL test cases. Breakout PASS and FAIL numbers for individual suites are also displayed. Best of all, the configuration window's summary tab facilitates quick navigation to an individual failure among possibly thousands of test case results.

Though it was not mentioned earlier, the exact source file and line number where a failure is logged is automatically and reported through CEV PassPORT's logging APIs. Since CEV PassPORT provides the source code for all of its test executables, being able to go directly to the source code reporting an error is a powerful adjunct to the textual descriptions of the failure.

Conclusion

Many aspects of the design of CEV PassPORT have been left out of this article. More can be written about how CEV PassPORT's architecture helps with the debugging of adaptation problems (for example, the test executables are debug builds and can be source level debugged). And though the descriptions above suggest how the system can be extended many of the particulars about the design decisions that went into CEV PassPORT to support rapid and painless extension have been omitted here.

CEV PassPORT is an integral tool of BSQUARE's QA process. Several existing customers are already taking advantage of CEV PassPORT's "positive feedback loop". User feedback about the system leads to enhancements in the user interface. And feedback about the provided tests leads to new subsystem tests as well as more robust tests with greater subsystem coverage. Most importantly, the ability of OEMs to provide stable and robust Windows CE adaptations on their devices will help make Windows CE the platform of choice for the next generation of smart devices.

About the Author

Jim Walters, Director, SmartBuild Software Development, has been with BSQUARE Corporation for four years. He has 18 years of experience in Quality Assurance and Software Engineering. In the past he has worked for or contracted to several large software houses, including Microsoft, Aldus, Adobe and Apple Computer.