Putting Your Software Through its Paces

Like all software products, AddonSoftware® by Barista® is an evolving complex system with many parts that make up the whole. The AddonSoftware business logic is the brains that control what it does, and the BBj®/Barista framework is how it gets that job done. With every change to either part of the system, the possibility of an error or incorrect logic entering into the system becomes greater. A robust testing procedure that uses tools that are consistent and logical finds such defects before they become big problems.

In traditional software testing, the developer performs the Component and Unit testing stages. Then the Quality Assurance team does System Integration testing, which can use up a significant amount of people, time, and money executing manual tests. Using an automated testing system is a better use of resources, exercises the software consistently, and finds defects more quickly. In fact, when an abnormal action or error happens, it can even automatically notify the testing team of a problem!

BASIS recognized the value of such a testing procedure and selected Quality First Software's QF-Test (www.qfs.de) for automated integration testing. QF-Test is a Java-based professional tool to automate testing of Java and Web applications with a graphical user interface, and it helps BASIS to find bugs faster.

Testing Infrastructure

A consistent software testing methodology in a modern programming environment provides critical feedback to both the developers and management that a quality product is being produced for the public. In a complex system like AddonSoftware by Barista, the chance of abnormalities showing up in even a well-designed, logically thought out design become even greater. Large or small, these defects need to be caught early in the development cycle to allow the developers time to correct them and to reduce the repair costs.

Using an automated testing framework such as QF-Test results in faster delivery of a cleaner product, a win-win for all. Customers receive a better product and BASIS can better use resources to address issues that do make it into the field.

Of course, there is nothing like the real world data and procedures to expose issues in the software that the engineers did not design for or anticipate. That is why BASIS includes sample databases in the downloadable BASIS Product Suite .jar file. Testing with the same set of consistent data in the databases allows BASIS to create and execute tests that can depend on predictable data and software responses. This allows QF-Test to execute tests on Barista and AddonSoftware designed to exercise as many of the key functions as possible with predictable results. When the results don't match what is expected, the test is flagged as having an abnormality to indicate that it needs to be examined further. Not all abnormalities are problems, some are minor but could be indicative of larger problems. In software testing, having as many eyes as possible on the product is a good thing. With the use of automated testing the process of finding problems before they are issues is much faster, more thorough, and repeatable. The result is a much more solid final product release.

Brian Sherman
Software Developer

Executing Test Suites

QF-Test controls the system under test by using test scripts called *Test Suites* to drive the testing sequences. A Test Suite is a collection of action steps that make up a logical sequence. Breaking up the steps into a series of *Test Case* steps makes it easier to debug the testing script when an unusual action occurs during execution.

Each Test Suite consists of a series of actions: Setup (start), the Test case execution, and Cleanup (stop). Each of these actions performs a critical role in the testing process. Setup is where QF-Test creates the test environment. Here, QF-Test sets the global variables, starts Barista, logs in, and takes control. Next, it starts to process the action steps of the tests. As it steps through individual tests, QF-Test checks for multiple conditions such as whether the forms and data display correctly, whether the prompts appear in the correct order, and if they appear within a reasonable time. If any of these conditions don't occur as expected, QF-Test stops execution and displays an error. If the Test Suite completes without any errors flagged, QF-Test executes the Cleanup section and shuts down the system under test.

The example shown in **Figure 1** runs through the AP to GL cycle. This Test Suite tests such key Barista components as Header/Detail Entry Grids, Document Processing (PDF and Jasper Reports), and of course the AddonSoftware business logic. It also tests for "under the hood" components such as displaying information, timing of actions, and operating system interactions such as cursor movement and keyboard function keys. As you can see, it logically steps through the business logic and performs the entry of an AP Invoice, selection and payment. With printing the 'Registers and Updates,' it verifies that the data flows through the AP module to the GL module.

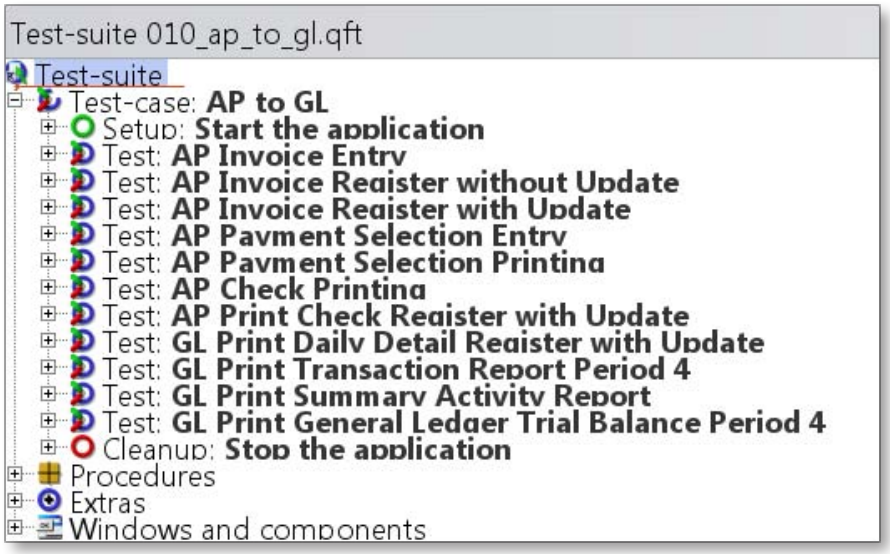

Figure 1. Test suites for AP to GL

Reporting Test Results

When QF-Test encounters an abnormality during execution of a Test Suite, it stops and brings up the display as shown in **Figure 2**. This shows a 'Component not found' type of error message window. In addition, QF-Test can automatically start its internal debugger sub-system to assist with tracking down the problem. This error occurred when the QF-Test was unable to properly start and connect to the AddonSoftware client.

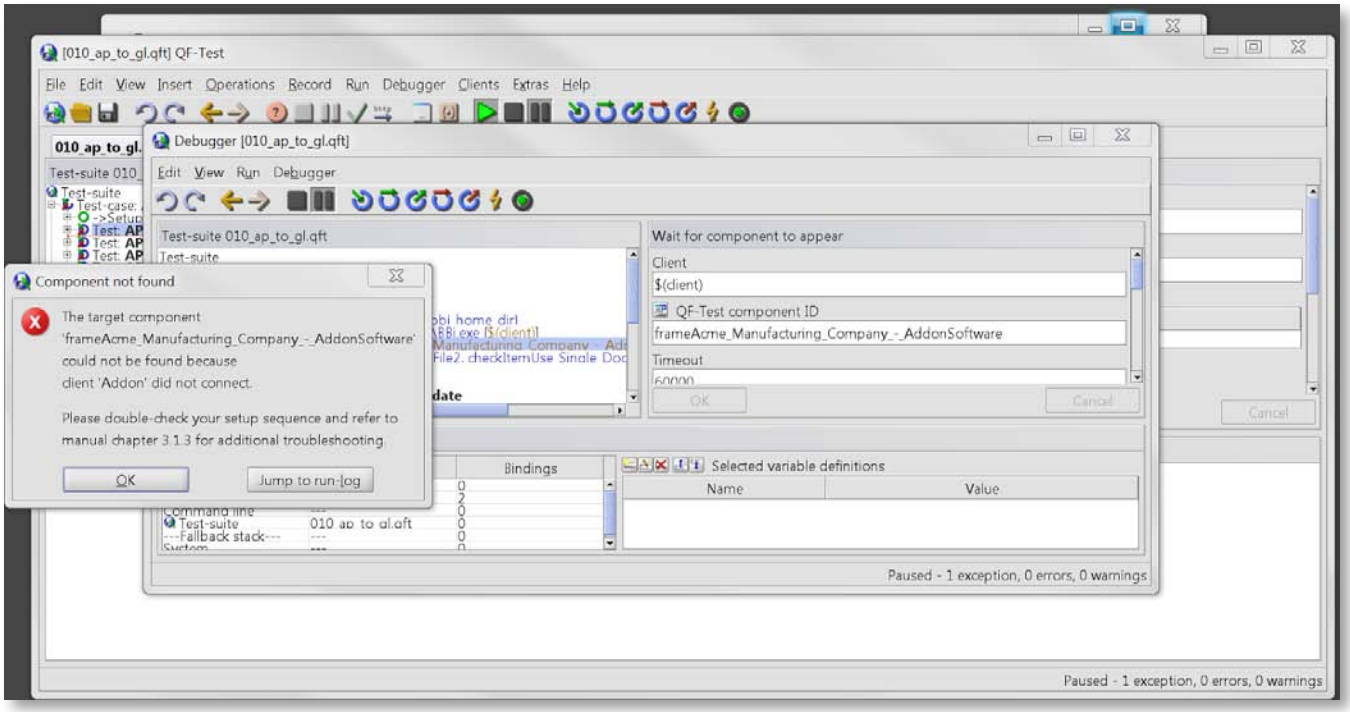

Figure 2. QF-Test 'Component not found' error

In **Figure 3**, QF-test has completed the Test Suite and reported '0 exceptions and 0 errors' as shown in the lower right corner of the Test Suite results. The number of warnings shown indicates that there were items that, while not an error or an exception, could cause problems and should be reviewed.

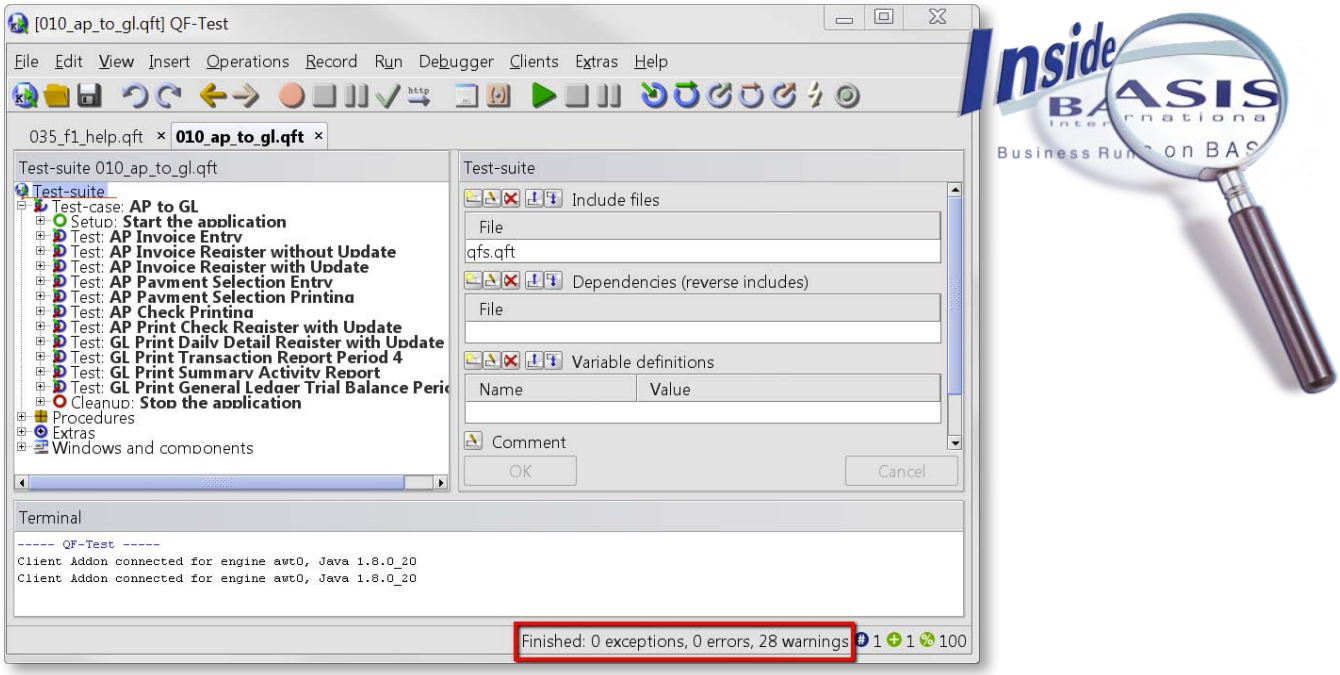

Figure 3. Successful Test Suite

To Test or Not to Test, That was the Question

With manual testing, there is the age old trade-off of resource costs against the quality gained. Deciding when to test was as much an art as a skill. However, with QF-Test's automated tests, we can run our tests as often as we like, kicking them off as a side-effect of a successful build, and have the results ready to view first thing the next morning. One benefit of this that often goes overlooked is that these same tests now work as Regression Tests. That means that we can run them on every build, regardless of what the development team has changed, and even detect side effects that may arise in unrelated areas of the product. In other words, we can make sure that we didn't break something in one place by fixing a defect in another.

Many Happy Returns on our Investment

Automated testing is not for everyone – there can be a considerable up-front cost in training and in developing the automated tests needed to cover your program adequately. Nothing is free. At BASIS, it is worth it. We do nightly automated builds and run the QF-Test tests on the result, only requiring human intervention when the test detected a defect. We call that "peace of mind," and it saves us money in the end. The sooner we detect a problem, the less expensive it is to fix, the faster we can release our products, and the sooner our developers and testers can move on to other important tasks. ■

> *"In particular, the testing of GUIs is more complex than testing conventional software, for not only does the underlying software have to be tested but the GUI itself must be exercised and tested to check for bugs in the GUI implementation. Even when tools are used to generate GUIs automatically, they are not bug free, and these bugs may manifest themselves in the generated GUI, leading to software failures."*
>
> Atif Memon, et al., Using a goal-driven approach to generate test cases for GUIs in 'Proceeding of the 21st International Conference on Software Engineering,' pages 257-266, IEEE Computer Society Press, 1999