NO MOCK OBJECTS FOR YOUR MAINFRAME? CREATE COBOL PROGRAM STUBS

Topaz for Total Test automatically creates and executes COBOL program stubs, allowing you to simplify testing efforts through program isolation.

Mainframe developers blazed such a trail in computer software, the dearth of mock libraries or COBOL program stub libraries today is surprising. In contrast, the open systems world provides many mock object libraries you can use to help isolate the systems required by a unit test.

Program stubs allow you to simplify testing efforts through program isolation. By calling program stubs rather than calling the actual programs, you can minimize the amount of code that must be tested at one time. It may require more tests, but each test will be more focused, and when a failure occurs it will be easier to discern the problem.

However, mainframe teams have long been forced to create their own libraries of COBOL program stubs for testing applications. The time spent maintaining these stub libraries for testing often carries a significant cost. The result is these libraries aren't maintained to reflect the changes to real programs, so they can no longer be used for testing. In other words, program stubs can go stale in environments where the real programs are changed frequently.

Mainframe teams need an easy way to create and execute COBOL program stubs, which would allow them to ensure this testing process becomes frequent and normalized, thereby increasing application quality. The solution is automated COBOL program stubs creation.

Automated Creation of COBOL Program Stubs

If a program already exists then you should be able to automatically create a program stub to simulate its behavior. This simulation works by returning data as though the program ran. Sometimes program stubs return static data when called and ignore the input parameters. More sophisticated program stubs can use the input parameters to determine the data to return. Regardless, program stubs do not include a complex implementation code and generally just return data.

It's relatively easy to change the data returned by a program stub. This capability is useful because getting a real program to return specific data can be complicated. When a program requires a specific error condition to trigger the needed data it can be time consuming so a program stub can really speed up testing of error conditions. Using a COBOL program stub, you can simply set up the program stub to return the data as though an error condition was encountered. This approach is often the easiest way to handle testing error conditions.

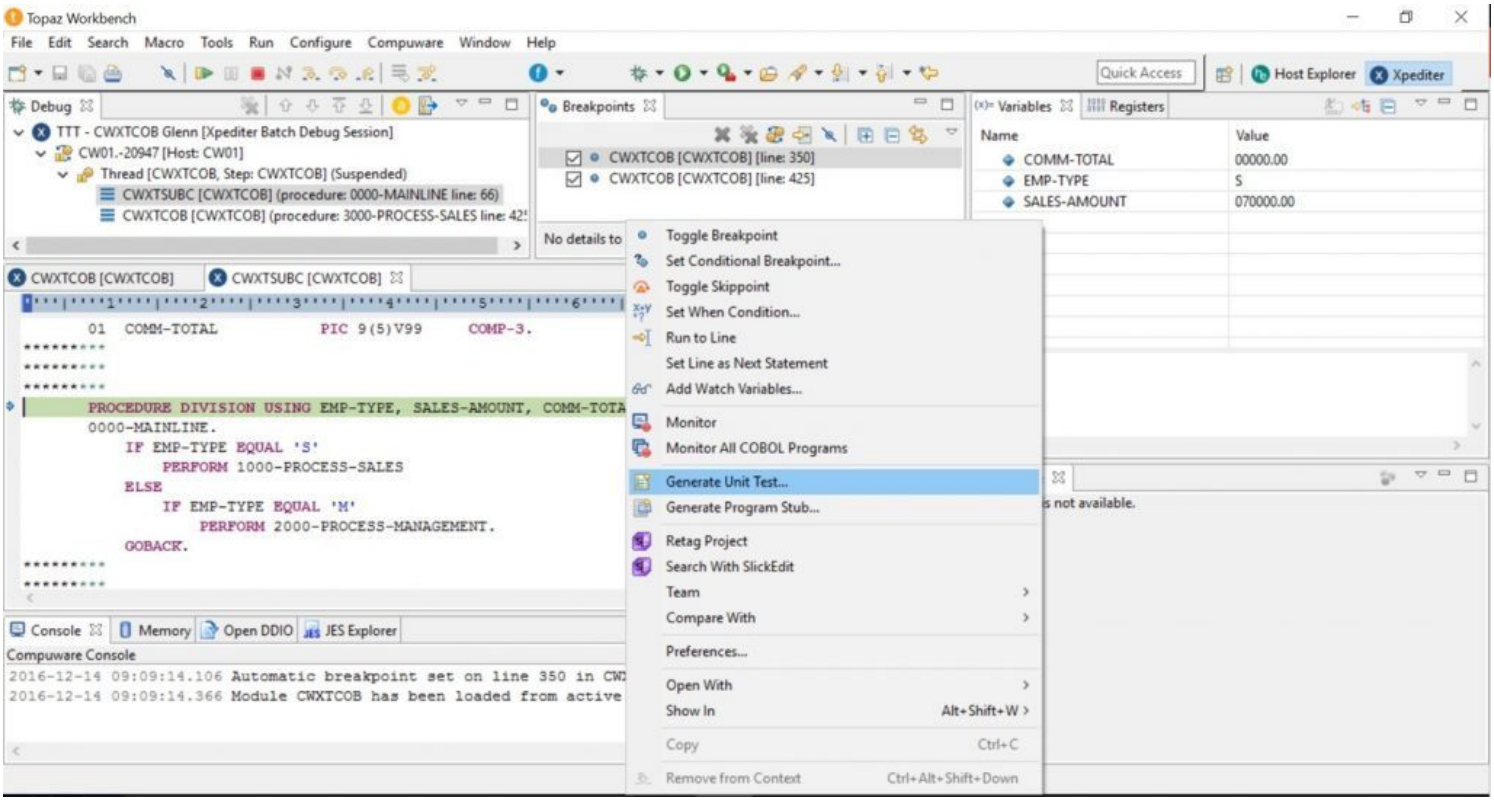
Testing error conditions is an area which is often neglected. If an error condition is handled poorly, it can have scary effects on the customer experience. One example could be a network error preventing a mainframe program from accessing a customer bank balance, so it just returns a balance of zero that appears on customers' mobile banking apps. The bank suddenly has a multitude of customers who are concerned about their bank accounts having a zero balance.

Program stubs can reduce the amount of work to set up a test environment. If you can stub out all the calls to another program or subsystem, you don't need to have it installed and working in your test environment. (Note that when you are doing integration testing you must have all the real programs installed and running; however, until you get to that phase of testing it can be much easier not to set everything up.)

Tools for Creating COBOL Program Stubs

BMC provides a solution for automating COBOL program stub creation with its BMC AMI DevX Total Test and Xpediter products.

Topaz for Total Test, an automated unit test creation and execution tool, can automatically create a COBOL program stub with data collected from Xpediter, AMI DevX interactive debugger, running your real program. As long as you use the latest version of your real program, you can get a completely up-to-date version of a program stub to use in your testing environments. A program stub can be created automatically in minutes for use with Topaz for Total Test.



Program stubs are also useful in situations where code is not yet complete or will be created by another party. Even if you don't have an existing program but you have an interface definition and know the data that will be returned, Topaz for Total Test provides a way to manually create a program stub so that you can debug, execute and test your program using a COBOL program stub in place of the unavailable code.

Even in the case where the code isn't written yet, Topaz for Total Test can simplify the process by allowing you to import a COBOL copybook definition of the interface for the program stub. Once this is done, it's a matter of entering the data to return from the program stub.

Today, your customers expect quality and velocity. Quality demands frequent testing, and velocity demands automation. Combining the two makes it possible to focus on other areas of development that lead to innovation but require more energy from your team. That's why a tool like Topaz for Total Test that automatically creates and executes COBOL program stubs, allowing you to simplify testing efforts through program isolation, is essential.