MARKET REQUIREMENTS DOCUMENT

Feature Name: Test Harness

Version: 3 **Date:** 4/13/04

Completed By: Leon Guzenda

Description of the Problem

We have written various specialized test harnesses to help QA our products over the years. We now have more platforms and languages and the product is too complex for the older test harnesses. We need a single "backplane" that we can plug tests into easily.

Description of the Requested Feature

Test Harness Capabilities

The test harness must be able to (at minimum):

- Run a sequence of tests synchronously.
- Stop a sequence if a test signals a fatal error or a designated time elapses.
- Run parallel sequences of tests.
- Stop all parallel sequences if any one of them signals a fatal error.
- Pause a test and have it wait for a set time, or for one or more named events to occur.
- Broadcast a named event to one or all parallel sequences. This could also be used to tell sequences to abort or shutdown.

An individual test must be able to:

- Pause for a set time, or wait for a named event from a named sequence, any other sequence or a set number of other sequences.
- Broadcast a named event to an individual or all parallel sequences.

Note that sequences may be running on different hosts.

Test Phases

For each test, or batch of tests, there will be five phases:

- An Initial Condition Setup phase.
- Action Phase (mixes of concurrent and sequential actions)
- A Final Condition phase.
- A Log Update phase.

- A guaranteed Cleanup phase.

Example - Suppose that the test is to create a new FD in the local directory, from scratch. The phases might look like this:

- Initial Condition Setup phase – List details of the environment. Run the guaranteed Cleanup script to remove all traces of the federation to be created and kill the lock server. Make sure there are no journals or recovery actions to be performed. Start the lock server. Ensure that the appropriate writeable directories are available.
- Action Phase: Run the oonewfd command with valid parameters.
- Final Condition phase: Check that the bootfile has the correct contents; that the FD file is there; that there is no journal file; and that oodumpcatalog runs and shows the FD to be there. If there is a tool to dump the internal schema then run it. Check that the FD file is writeable. If everything is OK return OK, otherwise return ERROR and an error number. Note that a test is deemed OK if it is supposed to produce an error response/message and it actually does.
- Log Update phase: If the test fails then record all appropriate information in a log file. If it is OK then record the fact that it ran OK.
- Cleanup phase: Remove all traces of the federation. Optionally, Kill the lock server.

Log Updates

Each phase of a test will write its identity, start date/time, test results and end date/time to a log file that is not in a directory that will be removed by the Cleanup phase.

Cleanup

Cleanup is to be done by a single script or method that is appropriate to the test(s) being run. The cleanup script or method must ignore non-fatal errors.

For example, the cleanup phase for catalog testing:

- Stops and deletes the lock server and any files that it may have created.
- Stops any AMS processes
- Kills any application processes and threads with a "Kill -9" command.
- Uses an operating system remove (rm) command to delete the boot file, federated database files (including those in othe partitions), database files, journal files, workfiles and any other external files.
- Deletes all directories that are used by the tests.
- Verifies that the directories no longer exist.
- If any file or directory still exists then force a failure of any sequence of tests that this phase is a part of.

Part of an existing feature or does it require another feature, if so, which one?

This facility is for internal use only.

How is this problem being solved now, and why isn't that acceptable?

We use a variety of mechanisms that are becoming increasingly hard to support across multiple platforms and languages.

What languages must this capability support?

- C++
- Java.
- Smalltalk
- SQL++
- Python

Which platforms must be supported?

- All currently supported platforms.

Do any competitors already have this feature?

- Not applicable.

Customers/Prospects who require this feature

- Not applicable.

Revenue at risk, or which could be won

- Revenue can decline to zero if product quality drops only a few percent.

When is this required?

- Release 9 or 10.

Additional Notes

1. Third Party Solutions, particularly grid-enabled ones, may be used to help implement this functionality.

2. We will also need a user manual for test developers.

3. Although not directly addressed above, the test harness should be able to direct human interactions as well as automated interactions, e.g. to request that a host be stopped, or to ask for input via a GUI.

4. The test harness can optionally interact with the build/QA software control system, but this is not mandatory. However, the Initial Condition Setup phase must be capable of logging details of its environment.

5. It is not desirable for the test harness to be linked with an Objectivity API. It is desirable to have a scripting language with Objectivity API, for writing tests that would be run by the test harness.

6. Command line arguments should consist of name & value pairs, e.g. –logfile mylog.

7. The test harness is responsible for maintaining configuration parameters and passing them to the test programs.