Case Study Information Appliance Test Plan

Overview

The following test plan describes the testing to be performed and/or managed by the "Some IA Maker" independent test team. It covers the included items in the test project, the specific risks to product quality we intend to address, timeframes, the test environment, problems that could threaten the success of testing, test tools and harnesses we will need to develop, and the test execution process. The independent test team is the quality control organization for "Some IA Maker". Some testing occurs outside of the independent test team's area, such as user testing and unit testing.

Bounds

The following sections serve to frame the role of the independent test organization on this project.

Scope

The following table defines the scope of the "Some IA Maker" independent test effort.

"Some IA Maker" Independent Test	
IS	**IS NOT**
Functionality (including client boot, client update, mail, Web, channels, etc.)	Usability or user interface (supporting role only)
Capacity and volume	Documentation
Operations (i.e., billing)	Code coverage
Client configuration	Security (handled third-party contract)
Error handling and recovery	Unit, EVT, or FVT testing (except for test system architecture)
Standards compliance (UL, FCC, etc.)	White-box/structural testing
Hardware reliability (MTBF, etc.)	
Software reliability (qualitative)	
Date and time (including Y2K) processing	
Distributed (leverage third-party labs and supplier testing)	
Performance	
Data flow or data quality	
Test system architecture (including unit, FVT and regression)	
Client-side and server-side test tool development	
Test database development	
Testing of the complete system	
Horizontal (end-to-end) integration	
Software integration and system test	
Hardware DVT and PVT test	
Black-box/behavioral testing	

Table 1: Independent test team IS/IS NOT (scope)

Definitions

The following table defines commonly used test terms and other terms found in this document.

Term	Meaning
Black Box Testing	Testing based on the purposes a program serves; i.e., behavioral testing.
Bug	Some aspect of the system under test that causes it to fail to meet reasonable expectations. "Reasonable" is defined by iterative consensus if it is not obvious.
Confirmation Test	A selected set of tests designed to find ways in which a bug fix failed to address the reported problem fully.
Driver	In this plan, a computer system running special software designed to generate traffic into a particular interface of the system under test. In particular, we intend to build a client driver and a server driver. (See Test Development below.)
Entry (Exit) Criteria	The parameters that determine whether one is ready to enter (exit) a test effort.
Integration Test	In this plan, a set of tests designed to find bugs in typical horizontal paths through integrated system components.
Oracle	A method or procedure, often integrated into a test tool, for determining whether the system under test is behaving correctly. This can involve examining all the outputs or a sample of the outputs, either from a user perspective (behaviorally) or from a system internals perspective (structurally).
System Test	A set of tests designed to find bugs in the overall operation of the integrated system.
Quality Risk	The possibility of a specific system failure mode, either localized, caused by subsystem interactions, or a knock-on effect of a remote system failure, that adversely affects the system's user.
Regression Test	A selected set of tests designed to find new failures, or regression, that changes, usually associated with bug fixes, have caused in subsystem, interface or product functionality.
SUT	System under test. In this case, the client hardware, the client software, the server software, the back-office operations, and the network infrastructure.
White-Box Testing	Testing based on the way a program performs its tasks; i.e., structural testing.

Table 2: Definitions

Setting

The test efforts described in this plan will take place in the following locations. (See Human Resources below for a description of the people in this table.)

Location	Test Effort	Staff
"Some IA Maker" (Austin)	Test tool development. Test case development. Integration and system test execution.	"Some IA Maker" Test Team
RBCS (San Antonio)	Test project planning. Test tool development. Test case development.	Rex Black (at "Some IA Maker" about 4 days/week)
Vendors	Testing of the software and hardware components provided.	Vendor Test Team Rex Black (process/results auditing)
[TBD—HW Test Lab]	DVT and PVT hardware test execution. (See Human Resources below.)	Test Lab Team Rex Black (process/results auditing)

Table 3: Locations involved in testing

Quality Risks

The following subsections define the risks to product quality that this test plan will help ameliorate.

Hardware

The client is subject to the same kinds of quality risks that exist for a personal computer. [Joanna/Alberto/Abdullah: Please help me prioritize these items, and point out any missing elements.]

Quality Risk Category	Specific Failure Mode	Priority
Reliability	Infant mortality Premature failure Battery develops "memory" Screen degrades (pixel death)	TBD
Radiation	Outside regulatory specs	TBD
Safety	Sharp surfaces Electrified areas Carpal tunnel/RSI Child/infant/pet issues Uncomfortable "hot spots"	TBD
Power	Brown-outs or transients affects unit "Noisy" or marginal power affects unit Excessive power consumption Electrostatic discharge	TBD

Quality Risk Category	Specific Failure Mode	Priority
Fragility	Moving parts (hinge, key, switch, contact, etc.) crack/break off Cracks/fails when dropped/bumped/slapped Loosens/breaks/warps/separates from shaking/vibration	TBD
Environmental	Fails under hot/cold/humid/dry conditions Unable to dissipate heat Fails/stains due to spills/spatters	TBD
Packaging	Inadequate protection of contents Hard to open	TBD
Signal quality	Bad I/O on external interfaces	TBD
Display quality	Bad pixels Contrast/color/brightness bad/inconsistent	TBD
Power management	Inadequate battery life Suspend/standby doesn't work/crashes	TBD
Performance	Slow throughput on modem (no 56K connects) Slow DSM memory access Insufficient CPU bandwidth	TBD

Table 4: Hardware quality risks

Because the item is an appliance, we anticipate the need for a higher level of ruggedness and forgiveness than is expected for a laptop computer. For example, resistance to spills and easy clean up are requirements. Drop tests will need to simulate typical indoor household surfaces such as carpet, tile, and linoleum.

Software

I analyze quality risks for the software at the system level. In other words, I consider the integrated system, including the client, the server, the PSTN, the ISP infrastructure, and the back-office capabilities. Quality risks pertain to certain conditions and scenarios that can arise in actual field situations.

Quality Risk Category	Specific Failure Mode(s)	Priority
Functionality	Client won't boot	1
	Client mail fails	1
	User data mishandled/lost	1
	Client browse fails	1
	Client software update fails	1
	Client user interface fails	1
	Device login fails	1
	Multiuser client logins fail	1
	Mail server loses mail	1
	Update server fails/unreliable/drops updates	1
	Preferences/state server fails/unreliable/loses state information	1
	Web server fails/unreliable/drops updates	1/2
	Scheduler/logging fails	1
	Sending mailer incompatibilities	1
	SPAM filtering ineffective	2
	Attachment refusal rude/silent	1
	RTF support fails/incomplete	1/2
	Basic/premium support differentiation fails	1
	Wrong number handling invalid	1
	Area code changes mishandled	1
	Reconnect time-out fails	1
Reliability	Client crashes frequently	1
	Server crashes frequently (w/o failover working)	1
	Net connection fails	1
Performance	Slow e-mail uploads/downloads	2
	Slow updates	2
	Slow Web access	2
	"Jumpy" action or intermittent slowdowns	2
Security	Denial of service	1
	Spoofing	1
	Mail data structure hacks	1
	Trojan horses	1
	Back-office hacks	1
	Internet hacks	1

Quality Risk Category	Specific Failure Mode(s)	Priority
Operations	Backup/restore on "live" server fails/slows system	2
	Update of software on "live" server fails/slows system	2
	Billing errors	1
	Incorrect customization (unit doesn't work/ads inappropriate)	2/3
	Administration/support fails	1
Error/disaster handling/recovery	No client recovery on bad/failed update	1
	Client crashes on PSTN disconnect	1
	DSM full crashes client	2
	No failover of server	1
	Server "death" unrecoverable	1
	Database corruption unrecoverable	1
Capacity/volume	Server slows/fails at/before 500,000 clients	2
	Infrastructure (ISP, etc.) slows/fails at/before 500,000 clients	2
	Server mishandles large e-mail	2
	Client mishandles large e-mail	2
	Clients crashes with full SDRAM (bad space management)	1
	Atypical usage profiles reveal choke-points	2
	Mailbox quotas	
Data flows/quality	Mail corruption	1
	Update package corruption	1
	Customer database corruption	1
	Incorrect accounting for usage	1
	Databases corrupted/inconsistent	1
Unreachable/stuck states	OS, app code, or app data (config) update gets "wedged" on server/client	1/2 [1] 1/2
	Mail not properly replicated to/from client	1/2
	Web page/content downloads/updates get stuck	1/2
	Timeouts/error handlers don't restore control to proper place	1/2
Untested code	Logic errors in untested branches/routines	3
Date/time handling	Time zones	2
	Y2K	1
	Leap years	3
	Daylight savings	3
	Internet atomic clock unavailable	3

[1] A wedged device is a serious problem, but wedged content may be okay.

Quality Risk Category	Specific Failure Mode(s)	Priority
Localization	Alphabet support problems Dial tone not recognized Power not tolerated Icons/logos/symbols incongruent with/unacceptable to local culture Keyboard/printer drivers not supported	Not tested
Client configuration options	Tethered problems Untethered problems Basic service problems Premium service problems	1 1 1 1
Documentation	Operations manual	1

Table 5: Software quality risks

Schedule

The following shows the scheduled events that affect this test effort.

[Gordon: This is the schedule you gave me, but it's still a bit "TBD" in terms of specifics until we have a budget and schedule approved. I'll update it at that time. Joanna/Gordon/Jennifer/Andrei/Alberto: What's missing?]

Milestone/Effort	Start	End
Test plan, budget, and schedule	4/21/01	5/7/01
FVT complete	5/24/01	7/12/01
Test team staffing	4/21/01	5/28/01
Lab Equip/Configure	5/4/01	5/28/01
EVT complete		6/1/01
Move to new offices	6/15/01	6/15/01
Dragon Boat Festival [no HW test to start/end]	6/17/01	6/17/01
Independence Day Holiday	7/2/01	7/4/01
Test driver development	5/4/01	6/15/01
Test suite development	5/11/01	7/12/01
Software integration test (two release cycles)	7/12/01	7/26/01
Hardware DVT	7/5/01	7/30/01
Software system test (five release cycles)	7/19/01	8/31/01
Hardware PVT	8/2/01	[TBD: Gordon/ Abdullah?]
General availability (GA)/First customer ship (FCS)	9/1/01	

Table 6: Scheduled milestones

Transitions

The following subsections define the factors by which project management will decide whether we are ready to start, continue, and declare complete the test effort.

Integration Test Entry Criteria

Integration Test can begin when the following criteria are met:

1. Two or more subsystems are ready to interact with each other on an actual or simulated (alpha) client device.
2. A server infrastructure, either production or simulated, exists to accept calls from the client devices and software

System Test Entry Criteria

System Test can begin when the following criteria are met:

1. Bug tracking and test tracking systems are in place.
2. The appropriate system administrator(s) configure the test network (see Test Configurations and Environments) for testing, including all target hardware components and subsystems, mail servers, update servers, Web servers, state servers, database servers, database tables (including indices and referential integrity constraints), network facilities, peripherals, firmware, operating systems, software, and related drivers. The Test Team has been provided with access to these systems.
3. The Development Teams provide revision-controlled, complete software products to the Test Team three business days prior to starting System Test.
4. The Test Team complete a three day "smoke test" and reports on the stability of the system to the System Test Phase Entry meeting.
5. The "Some IA Maker" Project Management Team holds a System Test Phase Entry Meeting and agrees that we are ready to proceed. The following topics will be resolved in the meeting:

   ✔ Whether all open design, implementation, and feature questions are resolved. For those question not resolved, the appropriate manager will commit to a closure date which will be no later than four (4) weeks prior to the planned System Test Phase Exit date.

   ✔ Whether all features are complete for the Device, Client, and Server. For those features not complete, the appropriate manager will commit to a completion date for the feature and for the feature's FVT/A-test which will be no later than three (3) weeks prior to the planned System Test Phase Exit date.

   ✔ Whether FVT/A-test is complete for the Device, Client, and Server. For those FVT/A-test efforts not complete, the appropriate manager will commit to a completion date for the feature and for the feature's FVT/A-test which will be no later than three (3) weeks prior to the planned System Test Phase Exit date.

   ✔ Whether all known "must-fix" bugs are addressed in the Client and Server software to be delivered for System Test. A "bug scrub" will be held. For any bugs not deferred or cancelled, the appropriate manager will assign target fix dates for all known "must-fix" bugs, which will be no later one (1) week after System Test Phase Entry.

- ✔ Whether all test suites and tools are complete. For any test cases and tools not complete, the Test Manager will assign target completion dates for each suite and tool not yet ready, which will be no later than three (3) weeks prior to the planned System Test Phase Exit date.

System Test Continuation Criteria

System Test will continue provide the following criteria are met:

1. All software released to the Test Team is accompanied by Release Notes.
2. No change is made to the Server or Client, whether in source code, configuration files, or other setup instructions or processes, without an accompanying bug report. Should a change be made without a bug report, the Test Manager will open an urgent bug report requesting information and escalate to his manager.
3. No change is made to the Device, whether in component selection, board layout, external devices, or the production process, with an accompanying bug report. Should a change be made without a bug report, the Test Manager will open an urgent bug report requesting information and escalate to his manager.
4. The open bug backlog ("quality gap") remains less than 50. The daily and rolling closure periods remain less than fourteen (14) days (all bugs are fixed within two weekly release cycles) once any initial pre-System Test bug backlog is resolved, which will be within one week. Twice-weekly bug review meetings will occur until System Test Phase Exit to manage the open bug backlog and bug closure times.

System Test Exit Criteria

System Test will end when following criteria are met:

1. All design, implementation, and feature question, code completion, and FVT/A-test completion commitments made in the System Test Phase Entry meeting were either met or slipped to no later than four (4), three (3), and three (3) weeks, respectively, prior to the proposed System Test Phase Exit date.
2. No panic, crash, halt, wedge, unexpected process termination, or other stoppage of processing has occurred on any server software or hardware for the previous three (3) weeks.
3. Production (B-Test or C-Test) Devices have been used for all System Test execution for at least three (3) weeks.
4. No client systems have become inoperable due to a failed update for at least three (3) weeks.
5. Server processes have been running without installation of bug fixes, manual intervention, or tuning of configuration files for two (2) weeks.
6. The Test Team has executed all the planned tests against the GA-candidate hardware and software releases of the Device, Server and Client.
7. The Test Team has retested all severity one and two bug reports over the life of the project against the GA-candidate hardware and software releases of the Device, Server and Client.

8. The Development Teams have resolved all "must-fix" bugs. "Must-fix" will be defined by the "Some IA Maker" Project Management Team.
9. The Test Team has checked that all issues in the bug tracking system are either closed or deferred, and, where appropriate, verified by regression and confirmation testing.
10. The open/close curve indicates that we have achieved product stability and reliability.
11. The "Some IA Maker" Project Management Team agrees that the product, as defined during the final cycle of System Test, will satisfy the customer's reasonable expectations of quality.
12. The "Some IA Maker" Project Management Team holds an System Test Phase Exit Meeting and agrees that we have completed System Test.

DVT Test Entry Criteria

DVT Test can begin when the following criteria are met:

1. EVT test has exited, and the Vendor has provided the Test Team with reports on testing performed, bugs found, bugs fixed, and bug remaining.
2. The Vendor has plans in place to resolve all known "must-fix" bugs.
3. Bug tracking and test case tracking systems are in place.
4. Final, approved versions of the primary specification documents (see Referenced Documents) exist and have been provided to test.
5. This document is final and approved.
6. An appropriate third-party test lab has been selected, a proper DVT test plan prepared and approved, and a contract signed with them.
7. The Vendor provides DVT units, in the quantities specified in this plan, for DVT test. (See Test Configurations and Environments.)
8. The "Some IA Maker" Project Management Team holds a DVT Test Phase Entry Meeting and agrees that we are ready to proceed.

DVT Test Exit Criteria

DVT Test will end when following criteria are met:

1. The HW Test Lab has executed all planned DVT Tests against the DVT units.
2. The Vendor has resolved all "must-fix" defects remaining from DVT or EVT test. "Must-fix" will be defined by the "Some IA Maker" Project Management Team.
3. The "Some IA Maker" Project Management Team holds a DVT Test Phase Exit Meeting and agrees that we have completed DVT Testing.

PVT Test Entry Criteria

PVT Test can begin when the following criteria are met:

1. DVT test has exited, and the Vendor has provided the Test Team with reports on their internal testing performed, bugs found, bugs fixed, and any bugs remaining.
2. An appropriate third-party test lab has been selected, a proper PVT test plan prepared and approved, and a contract signed with them.
3. The Vendor provides PVT units, in the quantities specified in this plan, for PVT test. (See Test Configurations and Environments.) [Joanna/Abdullah/Alberto: Do we

want the PVT units must produced on the production line or are engineering samples okay?]

4. The "Some IA Maker" Project Management Team holds a PVT Test Phase Entry Meeting and agrees that we are ready to proceed.

PVT Test Exit Criteria

PVT Test will end when following criteria are met:

1. The HW Test Lab has executed all planned DVT Tests against the DVT units.
2. The Vendor has resolved all "must-fix" defects. "Must-fix" will be defined by the Project Management Team.
3. The Test Team has checked that all issues in the bug tracking system are either closed or deferred, and, where appropriate, verified by regression and confirmation testing.
4. The open/close curves indicate that we have achieved product stability and reliability.
5. The "Some IA Maker" Project Management Team agrees that the PVT units, as tested during PVT Test, will satisfy the customer's reasonable expectations of quality.
6. The "Some IA Maker" Project Management Team holds a PVT Test Phase Exit Meeting and agrees that we have completed PVT Test.

Test Configurations and Environments

[Joanna/Gordon: We need to define the test lab. I haven't spent any time discussing this with people. Who are the right players? Where will it be?]
[Joanna/Abdullah: As first SWAG, per Joanna, we will have 50 DVT units and 100 PVT units. We'll need a pretty substantial number for the MTBF demonstration. Also, if we run an ALT, those tests will break the units, so we have to count on some number of samples being "consumed" by the testing.]
In case of anomalous behavior that nevertheless is not indicated as defective by a specification or other design document, the test engineer will repeat the test against all of the following reference platforms:

- IBM ThinkPad/Windows 98
- Dell Dimension/Windows NT
- Apple iMac/MacOS 8.0
- Micron Millenia/Linux
- Web TV

Each platform will run various versions of browser and e-mail software. [TBD: Amrita will provide browser list.] E-mail software will include, at a minimum, Yahoo, Hotmail, and AOL. We will use these to isolate performance, capacity, and functionality bugs on the "Some IA Maker" device. If the reference platforms display the same (defective) behavior as the "Some IA Maker" for any given test case, then the problem is mitigated.

Test Development

The Test Team will develop a complete set of test suites for Integration and System Test. We will work with the Client and Server Development Teams in an attempt to re-use and share test harnesses and cases with their unit and FVT efforts. However, because the

focus of these test efforts differs from that of Integration and System Test, economies are limited.

The tests developed will be both manual and automated.

The Test Team will also develop one or more test databases. They will load the appropriate test databases into the servers prior to running particular tests that require that data. For example, one database might hold all the customer information.

To test capacity, volume, performance, and reliability, we will need to simulate many thousands of users and long hours of client usage. Given the undesirability and impossibility of a manual approach, the Test System Architect and the Test Toolsmith will develop two drivers, one to generate traffic to the client, the other to generate traffic to the server. We will design these test tools as "dumb monkeys": they can generate tremendous amounts of data into the client and server interfaces, respectively, but do not include sophisticated oracles to verify correct behavior. However, each tool does include the ability to "stopwatch" certain activities for the purposes of measuring performance.

We will implement the client traffic generator on an x86-based PC running the Linux operating system. It will "talk" to the client through serial ports that will be provided through DigiBoard multiport boards. The tentative design is 48 ports per driver system. [TBD: Need to check Web site and verify drivers, but something like this will work. We will probably use a public-domain shell scripting language like TCL, which we can extend easily, to give us a programmable, interpreted interface. I'll need to put together a functional spec for this driver.]

[On the server side, we may be able to use some COTS test tools. For example, Silk, from Segue Software, can test Web sites. Also, "Some Mailing Software Vendor" should be able to provide us either with test tools or with direct test assistance to execute tests of the mail side. This would leave the update and state interfaces. We should be able to test this by talking to the servers over the LAN, via another x86-based PC running Linux. The test tool can initiate a download from the server by proclaiming its software or data level to be "below" the current revision. It receives the new release from the server but sends the data to /dev/null. The key challenge will be figuring out how to simulate load levels of 500,000 users. We may want to create some other server-side traffic generation tools that load specific subsystems, such as the database or the LAN infrastructure, that makes it "look like" 500,000 people are logged on, but we'd have to do some very careful performance modeling to figure out how to do that. Jennifer/Andrei, can we pull in the "Some Mailing Software Vendor" people, understand their piece of it, and then work together on a high-level design for the load generator(s)? Joanna/Gordon, I'm going to include some money in my proposed budget for COTS solutions on the Web side, if that's okay?]

We assume there will be no test development associated with DVT and PVT testing. The third-party test lab will supply the necessary hardware test equipment. We may need to provide simple scripts to exercise the system during operational tests, but those can be short QNX-based programs.

Test Execution

The following subsections define the participants and activities involved in test execution.

Human Resources

The table describes the human resources need to execute this plan. Key support and liaison roles are also defined.

Title	Roles	Name
Test Manager Test System Architect	Plan, track, and report on test design, development, and execution Secure appropriate human and other resources Design and implement test drivers Develop manual and automated tests Execute manual and automated tests Provide technical and managerial leadership of test team Audit test processes and results of external test participants	Rex Black
Server Test Engineer	Develop manual and automated tests Execute manual and automated tests Provide technical guidance and supervision to test technicians in the course of executing tests Work closely with Server Development Team	John Welsh
Client Test Engineer	Develop manual and automated tests Execute automated tests Provide technical guidance and supervision to test technicians in the course of executing tests Work closely with the Client Development Team	Junipero Serra
Test Toolsmith	Design and implement test drivers [Gordon/Joanna: This person can transition to a development role after these tools are created, provided he continues to support thee tools.]	Holly Watts
Test Technicians	Develop manual and automated tests Execute manual and automated tests [Gordon/Joanna: These are low-wage "button pusher" types with a couple years of test experience. I intend to get these guys from a staffing company I've worked with before. Also, we can put an ad in the Statesman and get some sharp college students. Let me know.]	Bob Lee TBD [3 people]

Title	Roles	Name
Unix System Administrator	Provide support for hardware and software installation in the Unix and QNX environments Assist with bug isolation Troubleshoot Unix- and QNX-related problems	Petra Persephone Luis Hernan Jack White (backup)
QNX System Administrator	Provide support for hardware and software installation in the Unix and QNX environments Assist with bug isolation Troubleshoot Unix- and QNX-related problems	Lilly Cohn Jack White(backup)
Network Administrator	Provide support for connection of the server and client systems to their various network (LAN, WAN, and PSTN) interfaces Assist with bug isolation Troubleshoot network-related problems	Lilly Cohn Jack White (backup)
Database Administrator	Install, configure, and repair databases on the test networks Assist with bug isolation Troubleshoot database-related problems	Luis Hernan Cindy Cruz (backup)
Hardware Liaison	Assist with bug isolation Troubleshoot hardware-related problems Route hardware-related bugs to appropriate people in the Hardware Development Team	Alberto Cordero
Client Liaison	Provide weekly builds during test execution Assist with bug isolation Troubleshoot client-related problems Route client-related bugs to appropriate people in the Client Development Team	Jennifer Lancaster (management) Zip Firenz (builds)
Server Liaison	Provide weekly builds during test execution Assist with bug isolation Troubleshoot server-related problems Route server-related bugs to appropriate people in the Server Development Team	Andrei Bronski (management) Zip Firenze (builds)

Title	Roles	Name
Network Liaison	Assist with bug isolation Troubleshoot network-related problems. Route network-related bugs to appropriate people in the Network Development Team	TBD [Gordon?]
Operations Liaison	Assist with bug isolation Troubleshoot operations-related problems. Route operations-related bugs to appropriate people	TBD [Gordon?]

Table 7: Human resources for testing

For hardware testing, we will leverage an external test resource to avoid having to outfit a hardware test lab. A third-party test lab can give us access to appropriate thermal chambers, shock and vibration tables, EMC/EMI radiation, and drop test equipment.

Test Case and Bug Tracking

Test cases will be tracked using a hierarchical collection of Excel worksheets. These will provide both detail level and summary level information on test cases.

Column Heading	Meaning
Priority	The priority of the test case.
State	The state of the test case. The possible states are: **Pass:** The test case concluded successfully. **Warn:** The test case concluded with an error, which the "Some IA Maker" management team has either deferred, closed as external to the product, or closed as unavoidable. **Fail:** The test case revealed a requirements or non-requirements failure that development will address. **Closed:** The test case previously revealed a failure that is now resolved. **In Queue:** The test remains to be executed (indicated by a blank in the column). **Skip:** The test will be skipped (explanation required in "Comment" column). **Blocked:** The test can not be run (explanation required in "Comment" column).
System Configuration(s)	A cross-reference to a worksheet that tracks specific configurations of client and server systems.
Bug ID	If the test failed, the identifier(s) assigned to the bug by the originator in "Some Bug Tracker".
Bug RPN	The risk priority number (severity times priority) of the bug(s), if applicable.
Plan Date	The planned date for the first execution of this test case.

Column Heading	Meaning
Actual Date	The actual date it was first run.
Comment	Any comments related to the test case, required for those test cases in a “Skip” or “Blocked” state.

Table 8: Test tracking

As the test organization runs each test, the state of each case will change from “In Queue” to one of the other states noted in Table 8: Test tracking.

For each test that identifies a problem and enters a “Fail” or “Warn” state, the tester will open a bug report in “Some Bug Tracker”. For each defect, “Some Bug Tracker” will track (at a minimum) the following: [TBD: Need to assess capabilities.]

Field	Meaning
Bug ID	A unique identifier for each bug.
Severity	Technical problem severity, as follows: **1**. Critical failure; **2**. Non-critical failure; **3**. Cosmetic; **4**. Suggestion.
Priority	The end-user priority of the issue or the item that failed: **Must-fix**. Highest priority, must-fix for release; **Candidate**. Medium priority, desirable but not must-fix for release; **Enhancement**. Low priority, fix as time and resources allow; **Deferred.** Not for this release.
System Revision(s)	The revision names for each system affected by or possibly involved in causing the error (see Release Management).
Subsystem	The system most affected by the error, from the following list: [I need to update this.]
Date Opened	The date on which the bug report was opened.
State	The state of the issue, as follows: [TBD: Fix this to correspond to “Some Bug Tracker”.] **Open:** The problem is deemed by the test engineer fully characterized and isolated; **Assigned:** The problem is accepted as fully characterized and isolated by development, and an owner, responsible for fixing the problem, is assigned; **Test:** Development have repaired the problem in some level of hardware or software, and someone owns testing the problem to evaluate the fix; **Closed:** The fix passed the test.

Field	Meaning
Originator	The name of the tester or other engineer who identified the problem, defaulting to the current user. For remotely generated reports, this will specify either the contact name or the outsource test organization itself.
Test ID	The test identifier (from the test-tracking matrix) corresponding to the test case the engineer ran that uncovered the issue. Also allowed are "Exploratory", "Other" and "Unknown".
Owner	The person responsible for moving the issue from "Assigned" to "Test" or from "Test" to "Reopened" or "Closed".
Abstract	A one- or two-sentence summary of the failure observed.
Remarks	A text field, free-format, consisting of two sections: **Steps to Reproduce:** A detailed, numbered process that will recreate the bug; **Isolation:** The steps performed to isolate the problem, including for reproducibility checking/statistics, bad-unit checking, and other pertinent tests.
Status	A free-from text field, consist of a "Comments" section and an "Actions" section, that updates what is happening to move the issue to closure (or deferral).
Closed Date	The date on which the issue was confirmed fixed or put on hold, which is used only when the issue is in a "closed" or "deferred" state.
Resolution	A free-format text field for a description of how the problem was resolved, which is filled in only when the issue is in a "closed" or "deferred" state.
Symptom	A classification of the symptom type from the standard "Some Bug Tracker" list.

Table 9: Bug tracking

When reporting a bug, test technicians and engineers shall adhere to the following process, taking detailed notes of all steps performed:

1. A bug report begins when a tester observes anomalous behavior on the part of the system under test, usually during the execution of the formal test case.
2. After reaching the logical conclusion of the failure, the tester repeats the steps required to reproduce the problem at least twice more, noting any variations in the behavior.
3. The tester then performs isolation steps as described below (see Bug Isolation).
4. The tester reports the bug, using "Some Bug Tracker", immediately after completing step three.
5. Prior to submitting the bug report, the tester finds a tester (preferably) or developer (alternatively) peer to review the bug report. Should the reviewer suggest any

changes or additional tests, the tester will perform those steps and add the details to the report. Once the report is considered satisfactory by both originator and reviewer, the tester submits the report.

Bug Isolation

In case of anomalous behavior, the test engineer will repeat the test against one or more of the following reference platforms:

- IBM ThinkPad/Windows 98
- Dell Dimension/Windows NT
- Apple iMac/MacOS 8.0
- Micron Millenia/Linux
- Web TV

Each platform will run various versions of browser and e-mail software. [TBD: Munira to provide list of browsers.] E-mail software will include at least Yahoo, Hotmail, and AOL. We will use these to isolate performance, capacity, and functionality bugs on the "Some IA Maker" device. If the reference platforms display the same (defective) behavior as the "Some IA Maker" for any given test case, then the problem is mitigated.

Release Management

During Integration and System Test, Test will receive builds, approximately weekly, from the Server and Client Development teams. While the server hardware will remain substantially unchanged, the client hardware will change during System Test. Integration Testing and the first two cycles of System Test will occur on DVT level hardware, while the final four cycles of System test will occur on PVT level hardware. On or around 8/1/01 (see Schedule) the test team will receive the PVT systems.

New releases come into the test organization either to add new functionality or to rectify a problem (reported as a bug or otherwise). In terms of test, five areas are important:

1. **Predictability and timing of releases.** Releases that show up at unpredictable times, too frequently, or too rarely can impede the forward progress of testing.
2. **Update apply process.** Ideally, the process of moving to a new release requires no external support, is simple, and is executable (via automation) by the test team in a matter of minutes.
3. **Update unapply process.** Sometimes bug fixes create more problems than they resolve, in which case a process for removing the update or recovering from the change is needed. Like the update process, simplicity and automation are goals.
4. **Naming.** When the test team reports new bugs, they need a way of identifying the offending releases. This requires a consistent naming convention for any subsystem release to test. The naming convention need not be meaningful, but it should imply sequence, as in A, B, C…
5. **Interrogation.** Naming conventions do very little good unless a tester has a way of determining the release names of all the subsystems by interrogating the system under test. This process needs to be as simple as possible to enable quick reporting of bugs.

In the first category, releases will come as described above. When it is necessary to accept an out-of-cycle release to keep testing moving, test will do so. The following table describes the other four elements of release management.

Subsystem	Apply	Unapply	Name	Interrogation
Server	Installed by Server Development Team with sniff test afterwards to ensure proper install.	As necessary, by the Server Development Team.	From the "Some Bug Tracker" build level.	Supplied to the test team by the Server Development Team upon completion of the install.
Client				
Application	chversion vernum devnum	chversion vernum devnum	Incremental based on application and OS; i.e., 189, 190, 191…	cat /version (via telnetd or local)
Operating System (QNX)	npcd src dst devnum	npcd src dst devnum	See above	[TBD: Jennifer?]
Device				
BIOS	[TBD: Alberto?]	[TBD: Alberto?]	[TBD: Alberto?]	[TBD: Alberto?]
Electronics	[TBD: Alberto?]	[TBD: Alberto?]	[TBD: Alberto?]	[TBD: Alberto?]
Enclosure	[TBD: Alberto?]	[TBD: Alberto?]	[TBD: Alberto?]	[TBD: Alberto?]

When the Hardware, Client, and Server Development teams provide new hardware or software to the test organization, they will also provide release notes. These release notes will, at a minimum, identify, by Bug ID, all bugs fixed in that release. A bug is considered "fixed" from a development standpoint when a developer has expended effort to resolve it and when he has run a unit test to confirm that the problem is indeed resolved. The release notes may be provided in a fax, e-mail, README, formatted document, or any other written fashion.
The release process, then, is as follows:

1. The developers notify the appropriate parties that their code and other files are ready for a build.
2. The appropriate parties prepare the weekly build. The build includes any file or configuration that influences the behavior of software on the client or server, not just those C and include files actually compiled and linked, but doesn't include content, e-mail, and other data sent to or received by the clients.
3. The build is released to test through an e-mail notification to all testers that a new build is ready. Release notes are delivered to the test team detailing any new features added and specific bugs (by bug ID) fixed in the release.
4. The test manager will notify the appropriate developers and testers when the current cycle of testing will end.
5. The test manager will stop testing and begin the installation of the new build on the test hardware.
6. For client builds, the test team will install their own clients. For server builds, the test team will authorize the appropriate developer to install and configure the appropriate server. The developers will each notify the server test engineer as they finish this process.
7. Once the last client and/or server are loaded with the new build, the test team will begin the new test cycle.
8. Should the quality of the new prove unacceptable to continue testing, the test manager will halt testing and release the test hardware for an "unapply" operation, reverting to the previous release. Testing will resume against that release, while waiting for an out-of-cycle build to repair the blocking problems.

Test Cycles

Test cycles will begin with each release of new client and server software. Test execution will begin as soon as the new releases are appropriately installed. Testing will be carried out in the following order:

- **Confirmation testing.** The test team will retest all bugs reported as fixed in the release notes. If fixed, the tester will close the appropriate bug report(s).
- **Scheduled testing.** The test team will run all test suites scheduled for that test cycle.

Should the test team complete all scheduled test suites, it will perform any other test suites not scheduled for that cycle but which have not been run against the release of software and hardware currently in the test lab. If those suites are then completed, the test team will perform exploratory testing until the following release begins the next

cycle. Conversely, due to delays of deliverables, schedule pressures, high bug find rates, or the size (duration or effort) of the test suites, it may not be possible to complete all the scheduled test suites during every cycle. In that case, the test suites not completed will be rescheduled as first priority for the next cycle, immediately following the confirmation testing.

Risks and Contingencies

The following table describes the key risks to success of this plan, and contingencies to address them.

Risk	Contingency
Unable to staff test team on time.	Reduce scope of test effort in reverse-priority order.
Release management not well-defined, resulting in a test cycle's results being invalidated.	Define a crisp release management process.
Test lab setup delayed or incomplete.	???
Test environment system administration support not available or proficient.	Identify system administration resources with pager/cell availability and appropriate Unix, QNX, and network skills.
Test environment shared with development.	[Joanna/Gordon/Jennifer/Andrei: I'd rather not do this, but I don't have a real good feel for what we're going to do in terms of lab space and hardware. Let's talk.]
Buggy deliverables impede testing progress.	Complete unit, EVT, and FVT testing. Adherence to test entry and exit criteria. Early auditing of vendor test and reliability plans and results.
Test and product scope and definition changes impede testing progress.	Change management or change control board.
[Joanna/Gordon: Others?]	

Table 10: Risks and contingencies

Change History

The following table outlines the change history for this document.

Revision	Released	Description/Changes	Author/Editor
0.1	4/26/01	First draft with extensive questions/ feedback requests to "Some IA Maker" team.	Rex Black
0.2	5/18/01	Second draft with many issues resolved, others identified.	Rex Black

Table 11: Change history

Referenced Documents

[Gordon/Joanna/Abdullah/Jennifer/Andrei: We need to reference all available specs, as well as making sure that I and the rest of the test team have access to all of them.]

This test plan content complies with ANSI/IEEE Std 829.