

COMPANY NAME

---

**Quality Engineering Standards and Practices**

COMPANY NAME

**Automation Plan  
Quality Engineering**

# COMPANY NAME

---

## Quality Engineering Standards and Practices

---

### SIGN-OFF SHEET

---

The following approvals are required to initiate the CompanyName Quality Engineering Automation Plan document for all future quality engineering projects:

Approval Name:	Department	Approval Signature:	Date Approved:

Other Project Participants for Distribution Only:

- Development Team
- Quality Assurance Team
- Project Managers
- Program Managers

# COMPANY NAME

---

## Quality Engineering Standards and Practices

<b>SIGN-OFF SHEET</b> .....	<b>2</b>
<b>PREFACE</b> .....	<b>4</b>
PURPOSE .....	4
AUDIENCE .....	4
REVISION HISTORY .....	4
<b>1.0 DESCRIPTION</b> .....	<b>5</b>
<b>2.0 DELIVERABLE DOCUMENTS</b> .....	<b>5</b>
<b>3.0 AUTOMATION APPROACH AND STRATEGY</b> .....	<b>6</b>
3.1 GENERAL APPROACH.....	6
3.1.1 <i>Smoke Test Automation</i> .....	9
3.1.2 <i>Performance/Stress Test Automation</i> .....	9
3.1.2.1 Scalability .....	9
3.1.2.2 Access Methods.....	10
3.1.2.3 Uptime and Availability .....	10
3.1.2.4 Security.....	10
3.1.2.5 Competitive and Industry Benchmarks.....	10
3.1.2.6 Load Balancing.....	10
3.1.2.7 Application and Web Servers .....	10
3.1.2.8 Database .....	10
3.1.2.9 Stress Testing .....	10
3.1.3 <i>Automation of Functional Regression Testing and Timing/Metrics</i> .....	11
3.1.3.1 Functionality Strategy.....	11
3.1.3.2 Metrics and Timing .....	12
3.1.4 <i>Automation Maintenance</i> .....	12
<b>4.0 AUTOMATION INFRASTRUCTURE</b> .....	<b>13</b>
4.1 OBJECT ORIENTED DEVELOPMENT .....	13
4.2 SCRIPTING STANDARDS .....	13
4.3 MERGING SCRIPTS .....	15
4.4 AUTOMATED SCRIPTS – MANAGEMENT AND CONTROL .....	15
<b>5.0 REPORTING AUTOMATION TEST RESULTS</b> .....	<b>15</b>
<b>6.0 TEST TOOLS</b> .....	<b>15</b>
<b>7.0 RESOURCES AND MANAGEMENT</b> .....	<b>16</b>
<b>8.0 ASSUMPTIONS, RISKS AND CONTINGENCIES</b> .....	<b>17</b>
<b>APPENDIX A – GENERIC CLIENT SYSTEM ARCHITECTURE</b> .....	<b>18</b>
<b>APPENDIX B – AUTOMATION SYSTEM ARCHITECTURE</b> .....	<b>19</b>
<b>APPENDIX C – AUTOMATION STRESS TEST LOG SAMPLE</b> .....	<b>20</b>

# COMPANY NAME

---

## Quality Engineering Standards and Practices

---

### PREFACE

This is the Quality Engineering Automation Plan document.

#### **Purpose**

The purpose of this document is to describe CompanyName's quality engineering standards and practices.

#### **Audience**

This plan is intended for use by the Engineering- Quality Assurance automation team to manage QE automation projects and to ensure that a quality product is developed before installing in a production client environment.

#### **Revision History**

Version	Date	Who	Revision
V1.0	1/05/01	FAL	

# COMPANY NAME

---

## Quality Engineering Standards and Practices

### 1.0 DESCRIPTION

---

This document is to be used by Automation Quality Engineer. It is a guideline for managing all QA automation projects. You may add to this document, as required.

Related Documents:

- Engineering Requirements Document
- Functional Specifications
- Technical Specifications
- Quality Plan
- Project Schedule
- Project Performance Plan: Stress, Load and Functional Testing
- Test Cases

The purpose of this document is to provide guidance for the automation of CompanyName client products. This document will also describe the automation infrastructure and the approach to automating under the current system architecture.

### 2.0 DELIVERABLE DOCUMENTS

---

The following documents will be written and placed under Configuration Management and Document/Source Control.

- CompanyName Automation Plan
- Project Performance Plan : Stress, Load and Functional Testing
- Automated LoadRunner Test Scripts
- Automated WinRunner Test Scripts
- Client System Architecture
- Automation Project Schedule

---

## Quality Engineering Standards and Practices

---

### 3.0 AUTOMATION APPROACH AND STRATEGY

---

#### 3.1 *General Approach*

Generally, all automation projects will follow the below guidelines and is the overall strategy for the QA automation effort. For each project, a document will be generated to identify the expected automation deliverables for a specific client.

Test Director will be used for the automation test planning process. WinRunner will be used for automating the functional regression and stress testing effort. The primary purpose of WinRunner is to automate the regression testing effort and to save project time that is required to regression test a product, as new functionality is added. Automation of the functional regression effort will be done if project time and resources are committed. WinRunner can also be used to automate specific functional transactions that are labor intensive, but will require that the functionality to be automated is stable.

Test Director will also be used to automate the storage and retrieval of test cases and the reporting of statistics for test cases executed. QA test documentation will grow tremendously within the next few years and it is imperative that this tool be used to control the management of all the documentation that will be generated.

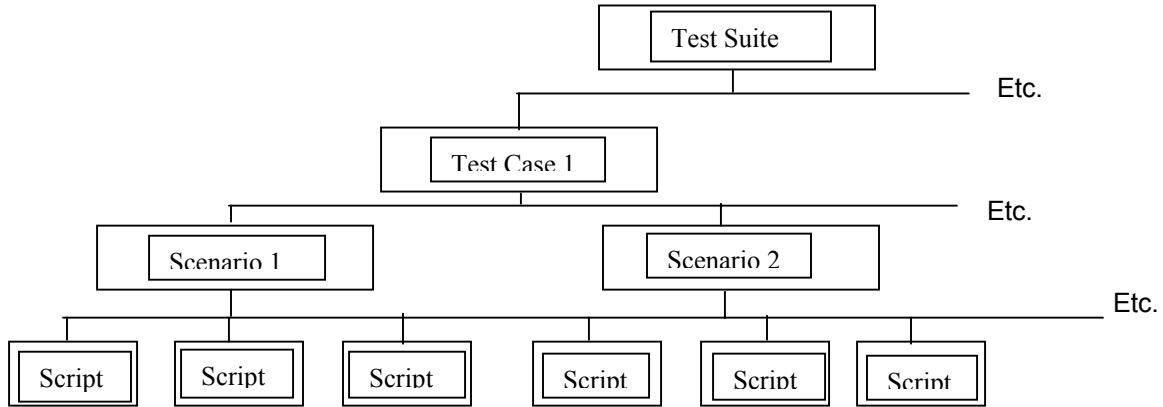
LoadRunner will be used to automate the load testing effort. This effort is important to CompanyName clients and the client's ability to handle a large volume of concurrent users.

In the future, a code coverage tool will be acquired to measure the effectiveness of the QA testing program by measuring the amount of code exercised, during the execution of automated test cases.

Automation will include the automation of the functional regression testing effort where possible and the automation of timing/metrics of certain transactions where possible. The writing of automated test scripts will be organized in the following manner. First a written test script is created, based on test cases previously written or manually executed. A script is a list of functions executed by the WinRunner and LoadRunner automation tools. A grouping of one or more scenarios for a certain feature is specified as a test case. A grouping of one or more test cases is specified as a test suite. This is a modular approach to automation. This type of approach will allow the QA Automation Team to maintain the development of future modifications in an object-oriented environment. This allows the QA Automation Engineer to modify a script in a specific area, instead of an entire Scenario.

---

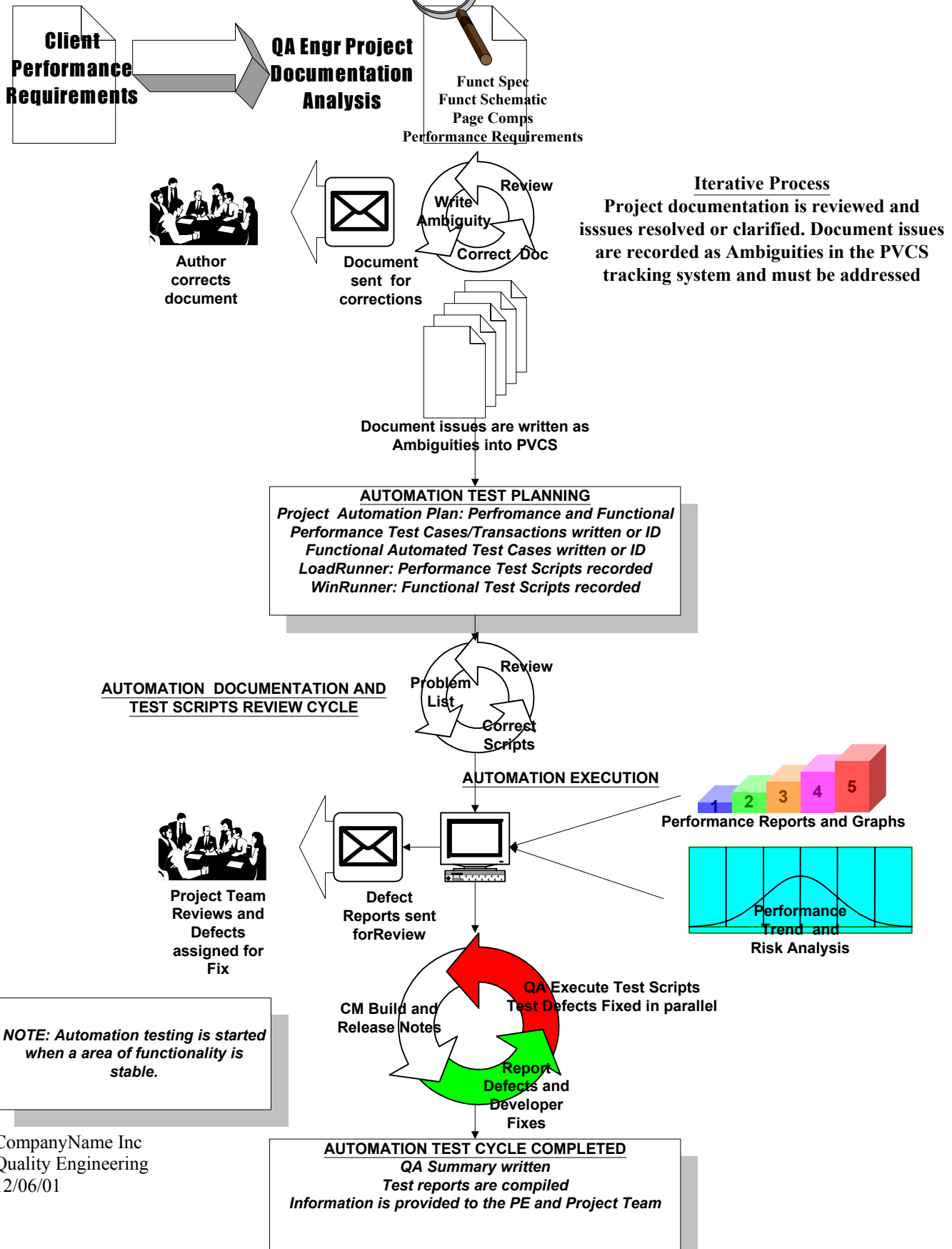
## Quality Engineering Standards and Practices



The automation of a specific client product and its related feature set are documented in the Project Automation Plan: Performance and Functional. This is the detailed documentation used to assist in the automation of specific features or functions. Refer to SQA Folsom directory under Methodology and Standards – Automation fro sample documentation.

The automation of the performance testing effort is separate automation project task. The purpose of this effort is to automate the performance of the product under stressful conditions with X number of concurrent virtual users. These virtual users are accessing the system simultaneously and exercising various functions and transactions for the product or feature under test.

Quality Engineering Standards and Practices



---

## Quality Engineering Standards and Practices

### 3.1.1 Smoke Test Automation

The purpose of the automated smoke test is to validate the basic and most important parts of the application. Making sure that things that were working on the previous build are still working and that the new code introduced did not change core functionality. This automated smoke test can be used as a functional baseline to build additional functional scripts. To be more efficient when coding the smoke tests, the automation engineers will request a list from the Project Engineer to identify the functionality they consider to be the most basic and essential features of the product under test. If time is allowed this task will be executed as a part of the automated project plan. WinRunner will be the tool for this automation effort.

### 3.1.2 Performance Load Test Automation

Load testing will be planned in the following increments (concurrency): 10, 20,30, 40 and upward to 250. Baseline data will be generated based on current client expectations. The baseline data will then be used to measure adequate performance guidelines. LoadRunner will be the tool for this automation effort.

The following sections are some (not all) of the areas that the performance project will focus on.

#### 3.1.2.1 Scalability

Performance testing strategy will require updating due to the growth of the client site and the functional growth of the product. This will be an on-going challenge and will be planned as the client requirements get expanded for future projects.

- **Test Transactions per second:** This test comprises the complete transaction across the client product, as a customer will traverse and interact with the functionality of the application. It measures the total number of transactions per second the system completed across a number of different users.
- **Test Kilobytes (KB) per second:** This script will test the amount of data exchanged back and forth between the customers connected and the Web server.
- **Test Round Trip Time:** This test is a measurement that acts like a stop watch that starts when a browser launches its first request in a transaction and stops when the browser receives the data from the last request. It is a measure of the end-to-end performance of the client application.
- **Test Concurrent Connections:** A request from a browser opens a connection with the Web server. Concurrent connections shows the number of connections to the Web server, depending on the number of users executing requests concurrently. This test will eventually reach the maximum number of connections that the Web server can handle.
- **Test Browser Page Download Times:** Static Web pages, Web pages that use Java and dynamically generated HTML pages.
- **Test Web server(s):** Test requests for static HTML. Test requests for web pages that need to be dynamically generated.
- **Test Application server(s):** Load test first in isolation, then end-to-end and monitor CPU and memory activity. If possible, test load balancing capability (test environment must replicate production).

## Quality Engineering Standards and Practices

- **Test Database:** Test maximum number of connections that the system will allow (make sure number of licenses are available for the test). Load test number of transactions in isolation, not end-to-end.

### ***3.1.2.2 Access Methods***

Transaction performance metrics for customer access through various modem speeds, T1, etc., will be based as indicated in section 3.1.2.1 above.

### ***3.1.2.3 Uptime and Availability***

Memory leaks related to the degradation of performance will not be covered in this plan. This is TBD.

### ***3.1.2.4 Security***

Security issues related to customer restrictions and how this is related to performance will not be addressed in this plan.

### ***3.1.2.5 Competitive and Industry Benchmarks***

Where required and available benchmarks will be acquired from competitive markets, so that the client will remain the leader in their industry.

### ***3.1.2.6 Load Balancing***

This type of testing would require heavy loads on the system under test to trigger balancing and the capability of triggering a load balance must exist in the test environment.

### ***3.1.2.7 Application and Web Servers***

Transactions will be executed at the Application and Web server(s) separately and end-to-end to determine transaction performance.

### ***3.1.2.8 Database***

Transactions related to database activity will be executed to determine database performance.

### ***3.1.2.9 Load Testing***

Performance load testing and monitoring strategy will be focused on the following areas:

- Client Web Server(s)
- Client Application Server(s)
- Client Database

---

## Quality Engineering Standards and Practices

In order to determine problem areas each of the above areas will need to be tested and monitored end-to-end and as separate entities. Testing and monitoring as separate entities will allow traceability of performance problem areas. Performance testing and monitoring end-to-end will allow the ability to analyze client products as a total functional system.

Initially, load testing of each separate entity and end-to-end will be done in the following manner:

- Concurrent customer log-ins
- Concurrent customer transactions per project
- Concurrent document distribution and types
- Concurrent Uploading/Downloading of different attachments
- Concurrent download, markup and upload different attachments
- Other performance test as mentioned above sections 3.1.2.1-3.1.2.9 will be conducted where necessary.

Performance requirements will be generated based on the customer acceptance levels currently being experienced in production. As the performance level of the client application improves, new metrics will be gathered and performance requirements upgraded.

NOTE: For more information on performance and functional testing for a specific project, refer to the Project Performance Plan: **Load, Stress and Functional Testing.**

### 3.1.3 Automation of Functional Regression Testing and Timing/Metrics

At the completion of automating a smoke test of the client product and/or feature, the client product and/or feature will be further analyzed and be entirely automated, where possible, to shorten the time frame it takes to execute a full regression test of the entire product and/or feature. WinRunner will be the tool used for this automation effort..

#### 3.1.3.1 *Functionality Strategy*

##### **a. Initial Operability**

This is primarily a smoke test of basic the client application functionality and will be covered under the automation plan to automate the smoke testing effort.

##### **b. End-To-End Functionality**

This is primarily a smoke test of the basic client application functionality as it related to significant customer transactions end-to-end (Example: Document distribution, a typical customer transaction workflow, etc.). Ideal scenarios will be collected from the Project Engineer. This test will also be included in the functional regression test suite, but with more functionality included.

---

## Quality Engineering Standards and Practices

### **c. Cross-Browser Compatibility**

This is primarily a smoke test of the basic client application functionality and the various browsers to be exercised (Netscape, IE and AOL). This test will also be included in the functional regression test suite, but with more functionality included.

### **d. Java Virtual Machines (JVM) Compatibility**

This is primarily a smoke test to ensure that Java applets work consistently across Java Virtual Machines (JVM). JVM for testing purposes will only be related to the various browsers and versions that the client utilizes. This will be covered under the automation plan to automate the smoke testing effort.

### ***3.1.3.2 Metrics and Timing***

Performance metric and timing involves the timing of a specific transaction or function. For example, timing the length of time it takes for x number of items to be listed on a log page. Another example, is the time it takes to upload and/or download a specific file type for a client product. Every function and or feature of a product needs to be analyzed to determine what parts can be automated for conducting this kind of test. It is of great benefit to manual test engineers to automate some of this manual testing effort early on and it may be done before the entire product or feature is stable. This may be throw away automation scripts that will only be used for the duration of the project. The automation of features for metrics and timing will considered at the time the project is in progress and the length of time it takes to automate. Considering the time frame for some projects, it may be faster to manually conduct this test than to automate it for the short term.

### **3.1.4 Automation Maintenance**

There will be periodic maintenance projects to update all of the current automated scripts and make them useful throughout the project lifecycle. Where possible automated scripts will be updated, as soon as the change as known, otherwise, maintenance projects will be scheduled on a regular basis.

---

## Quality Engineering Standards and Practices

---

### 4.0 AUTOMATION INFRASTRUCTURE

---

#### 4.1 *Object Oriented Development*

The concept of Object Oriented Automation is similar to the Object Oriented Programming (OOP) model. The idea is to facilitate the re-use and maintenance of code by breaking it into small pieces or functions (blocks). By creating small pieces of code for very specific functions, it is possible to create a function library of code, which is re-usable and will allow creating scripts faster and making them more flexible and easy to maintain.

Another important technique of Object Oriented Automation is the fact that variables and subroutines can be stored in INCLUDE files, and these files are made available to the entire application and/or test cases. By putting all the needed variables and major subroutines and functions in an INCLUDE file, time is saved whenever the values and/or data changes. Instead of having to revise every script making the call to the variable being changed, all that is needed is to change the value of the variable in the include file once. Now every time a script makes a call to this variable, it will collect the value from the INCLUDE file.

#### 4.2 *Scripting Standards*

##### **File Names**

To make it easier to identify different script files, they should be named in a consistent manner. The proposed criteria is to name files starting with the abbreviated project name followed by an under score and ending in a word or two describing the purpose of the script. For example: **emk\_registration.bdh**.

##### **Include Files**

If the include file will ONLY belong to one script, then it should be named following the convention set on the preceding section. Include files' names should represent the purpose of the INCLUDE file followed by the name of the Test Case/product. For example: [win\\_declarations\\_parade.inc](#)

##### **Subroutines Declarations**

Subroutines declarations should be included as much as possible in the INCLUDE files as this will help with the concept of Object Oriented (OO) programming.

##### **File's Headers**

Each automation script should begin with a generic header. This is used to inform the reader of basic information, such as: author, script's name, product and /or function being tested with it, etc.

Setup: This should include any information required in running the script. Such as, monitor resolution, special files (data files) needed to be used in conjunction with automation, etc.

---

## Quality Engineering Standards and Practices

Revision History: This should contain the date (mm/dd/yyyy), name of programmer/engineer to introduce the changes and a description of what has been added and/or changed and possibly, how this will affect the script.

```
=====
'
'AUTHOR: Engineer Name
'
'COPYRIGHTS: CompanyName, Inc. © 2001
'
'TEST CASE NAME: Basic Functionality
'
'TEST CASE PURPOSE: To verify basic features and functionality of the client application
'
'SPECIFIC AREA ADDRESSED BY THIS TEST CASE: Basic Functionality
'
'APPLICATION UNDER TEST: WHAM 2.1
'
'FEATURE UNDER TEST: Login application
'
'SETUP PREPARATION INCLUDING OTHER FILES REQUIRED TO RUN THIS SCRIPT:
'
'           1. Set display resolution to 1024x768
'           2. Close all directories and/or applications
'           3. Etc.
'
'REVISION HISTORY:
'           [1] mm/dd/yyyy - Engineer Name - COMMENT
'           [2] mm/dd/yyyy - Engineer Name - COMMENT
'
=====
```

### COMMENTS

Comments should be use through the code to explain difficult and or confusing areas of the script but not to the extent where EVERY line as a comment. Try to keep each comment to a few words or at the most no more than ONE line, unless, the code being explained is so complicated as to need such an extensive explanation.

Use the following style when commenting simple parts of code:

```
//Verify and click on Print button
```

When commenting complex parts of the code that will require more than one line use the box style:

```
//*****
//
//           COMMENTS GO HERE
//
//*****
```

---

## Quality Engineering Standards and Practices

To comment the beginning of a particular part of code use either one of the following styles:

```
//*****Login Test starts HERE*****
```

Or

```
//*****  
//*****LOGIN TEST*****  
//*****
```

### PROJECTS ORGANIZATION

Each new client product that enters the automation cycle should have its own workspace folder added in the automation directory. This workspace folder will be divided in the following subfolders: [documents](#), [results](#) (subdivided in: performance & functional), [LoadRunner](#) (subdivided in: source, include, results, userdata and workload) and [WinRunner](#) (subdivided in: inc, ini, opt, pln, res and t). As scripts are written for each part of the code, they should be added to their corresponding folders.

#### 4.3 Merging Scripts

The merging of new code to already existing scripts will be done by the QA Automation Engineers. The QA Engineers will be responsible for providing a one page report on their code, including some basic unit testing. Automation team leads will provide QA engineers with “temp” directories where to store their code until such time where they get merged and/or added to the main ones.

#### 4.4 Automated Scripts – Management and Control

The creation of automated scripts, both functional and performance, will be managed and controlled by the QA Automation Engineers. No member of the QA team should create new scripts, modify and/or append to existing ones without previous authorization from either of the QA Automation Engineers. Automation Engineers should keep each other informed of any code changes.

---

## 5.0 REPORTING AUTOMATION TEST RESULTS

Functional and performance results will be logged for every run done on a project in a Test Results table created by the Automation Engineer.

---

## 6.0 TEST TOOLS

The QA Automation Team will make use of the following technologies and tools for automation purposes.

### Microsoft – Word (Manual Test Cases writing tool)

---

## Quality Engineering Standards and Practices

Will allow the entire QA team to write testing documentation for the assigned project.

### **Visio – Visio (Software logic and mapping tool)**

With this tool, automation engineers and QA engineers will be able to graphically diagram the client system architecture and product data model.

### **Mercury Interactive - WinRunner (Functional Test Case automation tool)**

This tool will allow the automation engineers to create scripts by recording or manually coding them. Very powerful and expandable, it is ideal for an Object Oriented approach; also, it has a very extendable scripting language called TSL.

### **Mercury Interactive - LoadRunner (Stress, Load and Performance testing tool)**

Stress, performance and load testing scripts will be created, ran and monitored with this tool. This tool will allow the QA Automation team to closely monitor the status and performance of the client product. An important advantage of this tool is the capabilities of creating stress test where virtual users (vu) are created (machine and software dependent) to simulate multiple simultaneous connections/hits to the servers. In addition, it will allow pin pointing the processes that are stressing and killing the server.

### **Mercury Interactive – Test Director (Test Case Data Base)**

Test Director will allow the QA Team to store and retrieve test cases from a database. The amount of test cases written by the QA Team grows with each project. This tool will also allow the team to write test cases, execute them and report statistics and automatically.

---

## 7.0 RESOURCES AND MANAGEMENT

Resources for the automation effort will be assigned as required. A dedicated Automation Team must be put into place in order to make the automation effort a success.

Management will be done by weekly tracking the project schedule and collecting percentage complete from automation team members.

---

## Quality Engineering Standards and Practices

### 8.0 ASSUMPTIONS, RISKS AND CONTINGENCIES

---

1. QA resource allocation of projects for manual testing may impact the automation effort, when Automation Engineers are assigned to these projects. Automation will be delayed by the amount of days assigned to other projects.
2. If test data or the test environment is changed, then the automated scripts may need to be changed. It is important to keep these areas stable as possible.
3. Information regarding upgrades to current client technology needs to be communicated to QA Automation Engineers, otherwise the lack of information could impact the automation effort.
4. The availability of the test environment is critical to the success of the automation effort. When the test environment is unavailable, this will impact the project.
5. Critical fixes to current functionality makes the application unstable until tested manually by the QA Team.
6. A project client architecture diagram must be provided for each automation project requested.

### Appendix A – Generic Client System Architecture

TBD

## **Appendix B – Automation System Architecture**

Quality Engineering Standards and Practices

APPENDIX C – Automation Stress Test Log Sample

Test Type	Virtual Users	Duration	VU	Span VU	Start	Stop	Pass/Fail
Registration	100	30 min	25	300 Sec/0			
			40	300 Sec/5			
			55	300 Sec/10			
			70	300 Sec/15			
			85	300 Sec/20			
			100	300 Sec/25			
Navigation	100	30 min	25	300 Sec/0			
			40	300 Sec/5			
			55	300 Sec/10			
			70	300 Sec/15			
			85	300 Sec/20			
			100	300 Sec/25			
Registration and Navigation	100	30 min	25	300 Sec/0			
			40	300 Sec/5			
			55	300 Sec/10			
			70	300 Sec/15			
			85	300 Sec/20			
			100	300 Sec/25			
Registration and Navigation	100	20 min	25	300 Sec/0			
			50	300 Sec/5			
			75	300 Sec/10			
			100	300 Sec/15			
Registration and Navigation	100	1 Hour	25	60 Sec/0			
			50	60 Sec/1			
			75	60 Sec/2			
			100	60 Sec/3			
Registration and Navigation	100	8 Hours	25	60 Sec/0			
			50	60 Sec/1			
			75	60 Sec/2			