



Functional and Performance Test Plans

Kahala Posts Groups

June 2004

Table of Contents

INTRODUCTION3

TEST PHASES.....4

FUNCTIONAL TEST PLAN5

Methodology.....5

Test Scope7

Test Data.....8

Trouble Tracking8

PERFORMANCE/STRESS TEST PLAN.....9

Methodology.....9

Test Scope10

APPENDIX.....11

Appendix A – Functional Test Case.....12

Appendix B – Hardware Diagram.....13

Appendix C – Software Diagram.....14

Appendix D – Performance Monitor15

Appendix E – Trouble Tracking Form16

Appendix F – Definitions and Evaluations17

Client Machine17

Webserver (WAE – Static Content).....17

ApplicationServer (WAE – Dynamic Content).....17

Database.....17

Network.....17

Introduction

The Kahala Posts Group (KPG) project will streamline postal traffic between the U.S. and the Pacific Rim countries (Australia, Japan, Hong Kong, Korea, and China) by guaranteeing package delivery using a guaranteed date calculator.

The calculator will determine the travel days between postcode pairs using a series of API's (*Application Program Interface*), including: countries, provinces or states, cities, and country work days.

This document presents the testing plan that will validate the **functionality** and **performance** of the guaranteed calculator. It also describes the “trouble tracking” tool for reporting, prioritizing, resolving, and closing defects found during the test phases.

All testing will be performed by the development team at the USPS headquarters in Arlington, Virginia.

Note: This testing plan is comprehensive to the extent of current data, but not necessarily conclusive. Subsequent changes or enhancements may be needed to reflect modified functionality.

Test Phases

There are three phases of testing:

- 1) Unit - Tests individual component's input and output using **JUnit** (unit-testing framework). Unit-testing verifies the results expected from java classes or components
- 2) Functional – System and integration testing performed concurrently for system features using the **API's** that link to the calculator through the **SOAP** (*Simple Object Access Protocol*)
- 3) Performance/stress – Focuses on the behavior of the API's under the agreed upon load

Each phase will use test cases (*see Appendix A*) based upon production data from all countries associated with KPG. Successful completion of each phase is required before moving forward.

Functional Test Plan

Methodology

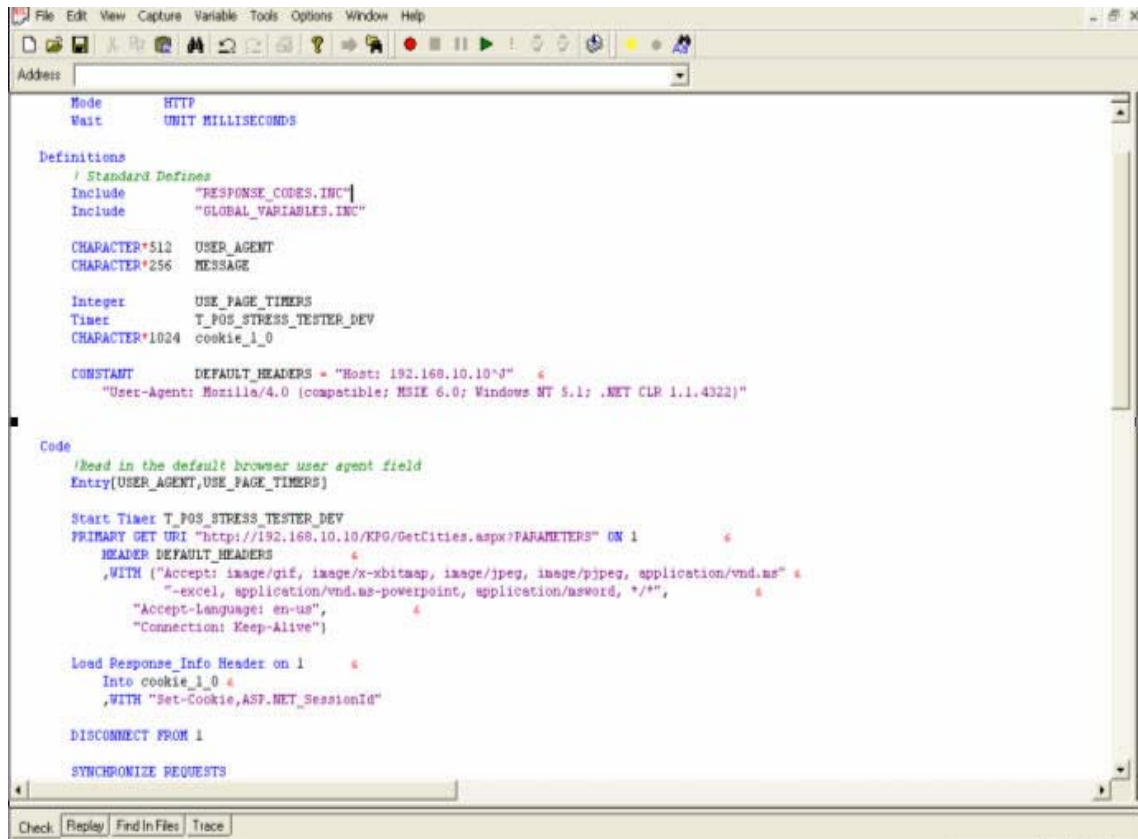
The functional test will use the the OpenSTA toolset to test the SOAP API components and functionality. (There is no GUI, *Graphical Unit Interface*, for this project.) OpenSTA is a distributed software testing architecture. It has the capability of performing scripted HTTP heavy load tests with performance measurements from Win32 platforms.

OpenSTA captures the http requests and responses from a SOAP interface. The HTTP response will be used to validate the API's functionality. Each SOPA API will have an HTML page for entering data. (The HTML are for testing purposes only, and used to start the SOAP API's.)

Results and statistics will be collected during test runs by a variety of automatic and user controlled mechanisms, and much of the data logged can be monitored live during test runs.

Note: OpenSTA is licensed and free, so there is no official time frame to resolve issues if they arise, and no maintenance or support costs. The OpenSTA toolset has been evaluated by the development team, and appears to be adequate and stable. For more information: www.opensta.org

Sample Functional OpenSTA



```
File Edit View Capture Variable Tools Options Window Help
Address
Mode HTTP
Wait UNIT MILLISECDS

Definitions
/ Standard Defines
Include "RESPONSE_CODES.INC"
Include "GLOBAL_VARIABLES.INC"

CHARACTER*512 USER_AGENT
CHARACTER*256 MESSAGE

Integer USE_PAGE_TIMERS
Timer T_POS_STRESS_TESTER_DEV
CHARACTER*1024 cookie_1_0

CONSTANT DEFAULT_HEADERS = "Host: 192.168.10.10" &
"User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)"

Code
/Read in the default browser user agent field
Entry[USER_AGENT,USE_PAGE_TIMERS]

Start Timer T_POS_STRESS_TESTER_DEV
PRIMARY GET URI "http://192.168.10.10/KFG/GetCities.aspx?PARAMETERS" ON 1
HEADER DEFAULT_HEADERS
,WITH ("Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/word, */*",
"Accept-Language: en-us",
"Connection: Keep-Alive")

Load Response_Info Header on 1
Into cookie_1_0
,WITH "Set-Cookie,ASP.NET_SessionId"

DISCONNECT FROM 1

SYNCHRONIZE REQUESTS

Check | Replay | Find In Files | Trace
```

Test Scope

The following API's will be tested. All test cases will be prefixed with "KPG" and the SOAP API name as the document name.
For example: KPG-GetCities-001.

API	Definition
Get Countries	Get a list of all KPG countries
GetStatesOrProvinces	Get a list of KPG States or Provinces
GetCities	Get a list of cities in this KPG State or Province
GetPostalCodesForThisStateOrProvinceNameAndThisCityName	Get a list of postal codes in this State or Province name and in this city
GetCitiesAndStatesOrProvincesCodeForThisPostalCode	Get a list of Cities and States or provinces names for this Postal Code
IsValidPostalCode	Is this a valid postal code
IsValidStateOrProvince	Is this a valid State Or Province
IsValidCityInThisStateOrProvince	Is this a valid City in this State or Province
IsCountryWorkDay	Is this a work day
IsCityWorkDay	Is this a work day for this city
IsHoliday	Is this a Holiday
GetGuaranteeDateInformation_UsingPostalCodes GetGuaranteeDateInformation_UsingPostalCodesStateOrProvinceAndTheseCities GetGuaranteeDateInformation_UsingPostalCodesAndOriginStateOrProvinceAndCity GetGuaranteeDateInformation_UsingPostalCodesAndDestinationStateOrProvinceAndCity	Get Guarantee Date Information Using Postal Codes (Simple search) Get Guarantee Date Information Using Postal Codes State or Province and these Cities Get Guarantee Date Information Using Postal Codes and Origin State or Province and City Get Guarantee Date Information Using Postal Codes and Destination State or Province and City

Test Data

Test data will come from a production load of Australia, Honk Kong, Japan, Korea, and the United States. Additional test data will be created to test the boundaries of each SOAP API. Data for all countries is needed to validate the multi-language testing requirements.

Trouble Tracking

A form has been designed to record “troubles” (problems) found during testing (see *Appendix E*). Each trouble will be assigned a fix-priority based upon the charts below.

Test cases will be considered completed when troubles with high and medium severities are resolved or have plans of resolution.

System Impact

Priority	Definition
High	Functionality testing cannot continue
Medium	Functionality testing can continue Medium effort on re-testing cases
Low	Functionality testing can continue Low effort on re-testing cases

Business Impact

Severity	Definition
Critical	Critical to the business No alternative paths to perform the tasks
High	High impact to the business No alternative paths
Medium	Feature rarely used no alternative paths
Low	Alternative paths available
No Impact	Cosmetic changes such as message type errors, colors, etc.

Performance/Stress Test Plan

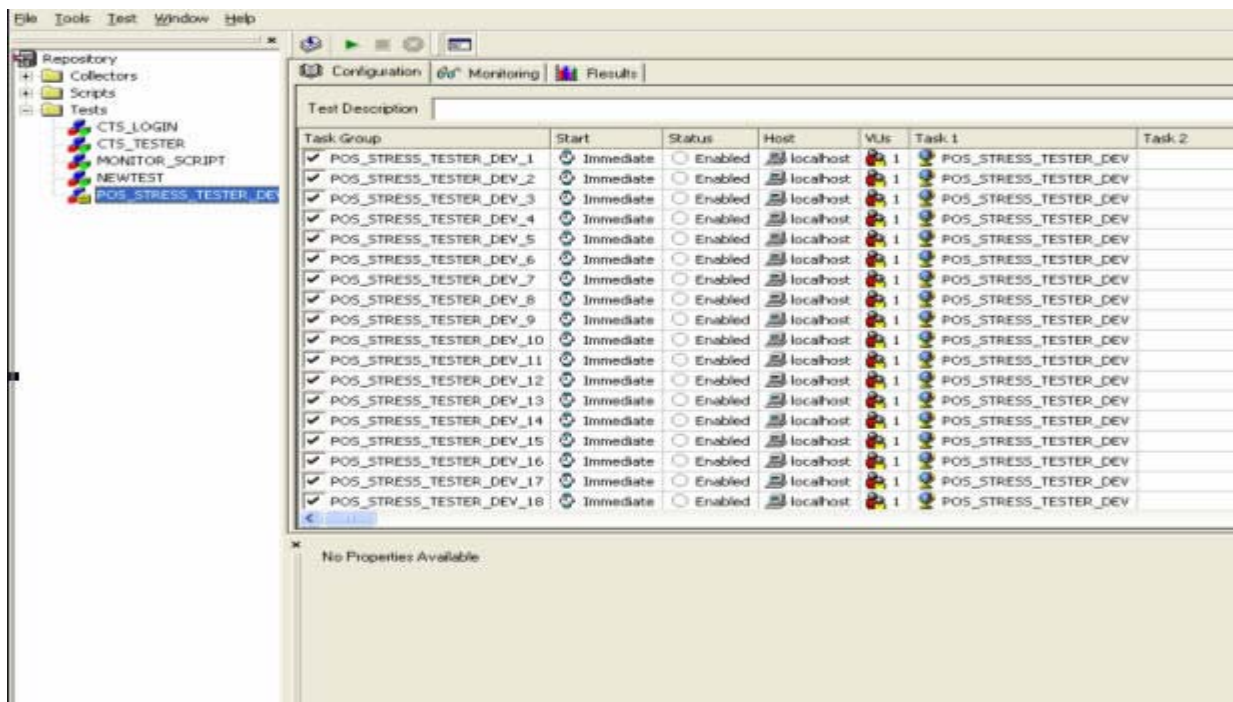
Methodology

Performance/stress testing will also use the OpenSTA toolset. The capabilities of the calculator software will be tested by simulating multiple user HTTP requests, and responses to the WAE (*Web Application Environment*) . “Real world” scenarios will be created using data compiled from the KPG participating countries.

Testing will determine if:

- 200 user hits per second can be supported by the current hardware and software architectures
- The application can handle the forecasted 10K response message

Sample Performance/Stress OpenSTA



Test Scope

Three performance measurements will be performed per test case. This will minimize the potential of the timings being skewed due to network delays. An average will be taken between the collected performance times, and the caching component will always be cleared before each performance run to simulate the worse case scenario.

- 1) Client Machine - There will be four machines used to run the scripts that will simulate 50 users per machine. The scripts will contain the following:
 - Retrieve countries
 - Retrieve states or provinces
 - Retrieve postal codes
 - Provide Guarantee Date Information using: postal codes; origin state/province and city; destination state, province and city
 - Provide validation on: postal code, state or province; state, province or city; country workday; and holidays
- 2) Application Serve (WAE) – Test cases will be logged during the performance stress test with:
 - Transaction start time
 - Transaction stop time
 - Thread identifier
 - Time stamp
- 3) Database – The efficiency of the SQL statements and the database performance will be monitored using the tools provided by Oracle.

Note: The performance assessment will not include any analysis on the client machines or on the network latency. The network architecture between the development and the production USPS environments is difficult to simulate and reproduce.

Appendix

Appendix A – Functional Test Case

Test Case

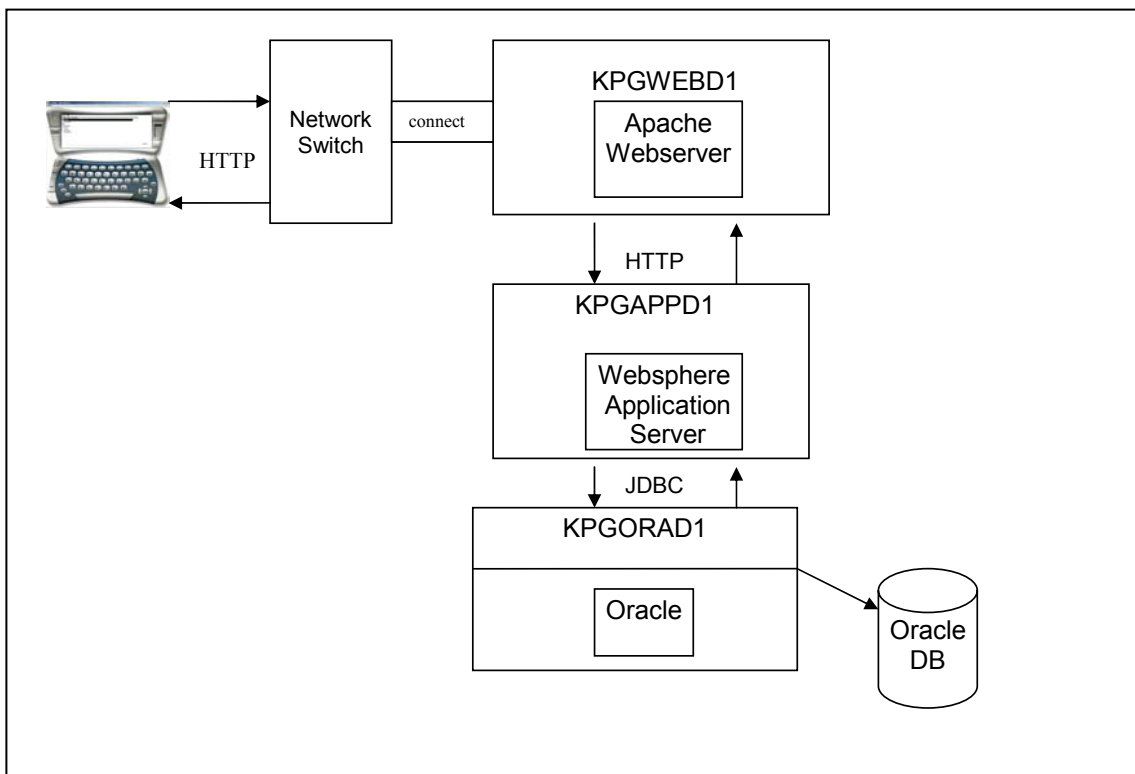
Test Cycle <01> - <GetCities API>			
1			
Test Case No – KPG-Get-Cities-001		Test Name – GetCities SOAP API one city	
Test Objective – Validate the GetCities SOAP API returns appropriate results if one city for the state or province is found on the database			
Prerequisite Tests – N/A		Date Tested –	
Dependent Tests – N/A		Tested By –	
Data/Configuration Requirements – A valid language code, country code, state or province code need to be provided. The state or province only has one city that is associated with it.		Time Taken –	
Step	Action/Inputs	Expected Results	Result (Pass/Fail)
	Send the following xml values: <pre> <soap:Body> <GetCities xmlns="http://www.kahalaPostsGroup.com/SOAPI NTERFACE" <LanguageCode>en-us</LanguageCode> <CountryCode>US</CountryCode> </GetCities> </soap:Body></pre>	SOAP API GetStateOrProvinces Returns: <pre> <?xml version="1.0" encoding="UTF-8"?> <GuaranteeCalculatorCitiesReturnMessages Version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.kahalapostsgroup.com/gc/Schema/GCCiti es.xsd"> <Messages NumberOfRecords="1" LanguageID="en-us"> <Message> <MessageNumber>1</MessageNumber> <MessageDetail>100</MessageDetail> <MessageDescription>Successful</MessageDescription> </Message> </Messages></pre>	

Appendix B – Hardware Diagram

Hardware - There are 3 machines involved in testing the KPG SOAP API's:

- a) **Client machine** – Where the testing is done
Pentium 3 1.7 GHz 512 Megs of RAM and 40 Gig hard drive.
Internet Browser is IE 6.0 XP SP2 running under Windows XP 5.1 SP1.
- b) **Webserver and Application Server machines** - KPGWEBD1 and KPGAPPD1
Compaq Proliant DL380 G3 Servers with 2 (two) Pentium 3
2.4 Ghz processors with 1580MB of RAM.
146 GB hard drive. Windows 2000 Server SP4. Apache is
the webserver, and IBM's Websphere 5.0 is the application server.
- c) **Database machine** (KPGORAD1) – Where the data (persistent) resides
Compaq Proliant DL380 G3 Server.
Two Pentium 3 2.4 Ghz processors with 1580MB of RAM.
Two 18GB hard drive.
Page file is set for 2046MB-4092 MB.
OS is running
Windows 2000 Server SP4, an Oracle 9I Dabatase Server is the data
server.

Hardware Components Diagram

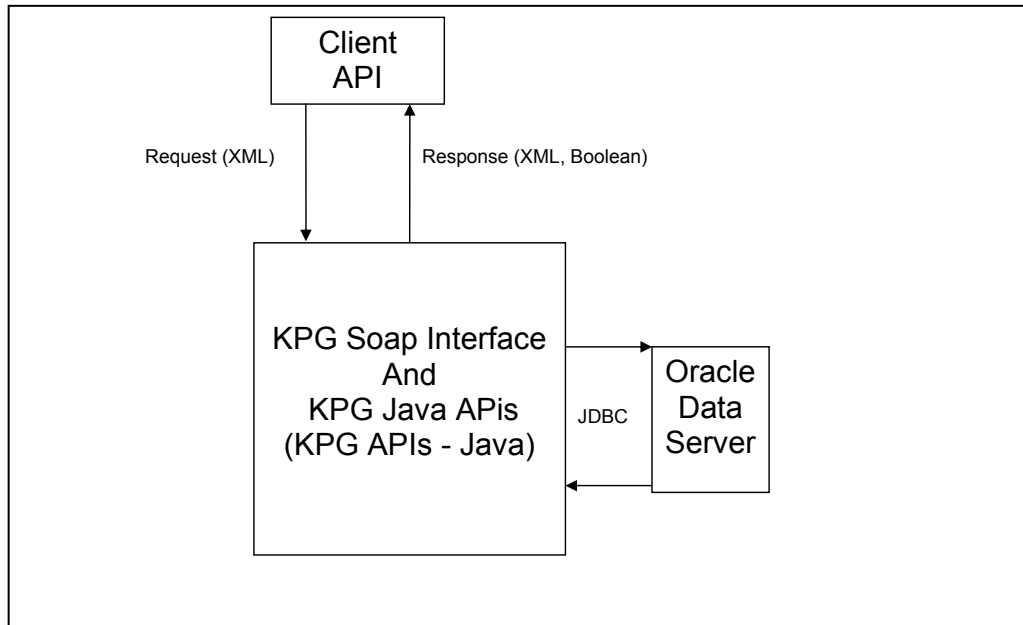


Appendix C – Software Diagram

Software:

There is no need to purchase any automation testing tools. The project testing activities will revolve on the OpenSTA toolset. During the OpenSTA evaluation period, the toolset demonstrated flawless testing operations.

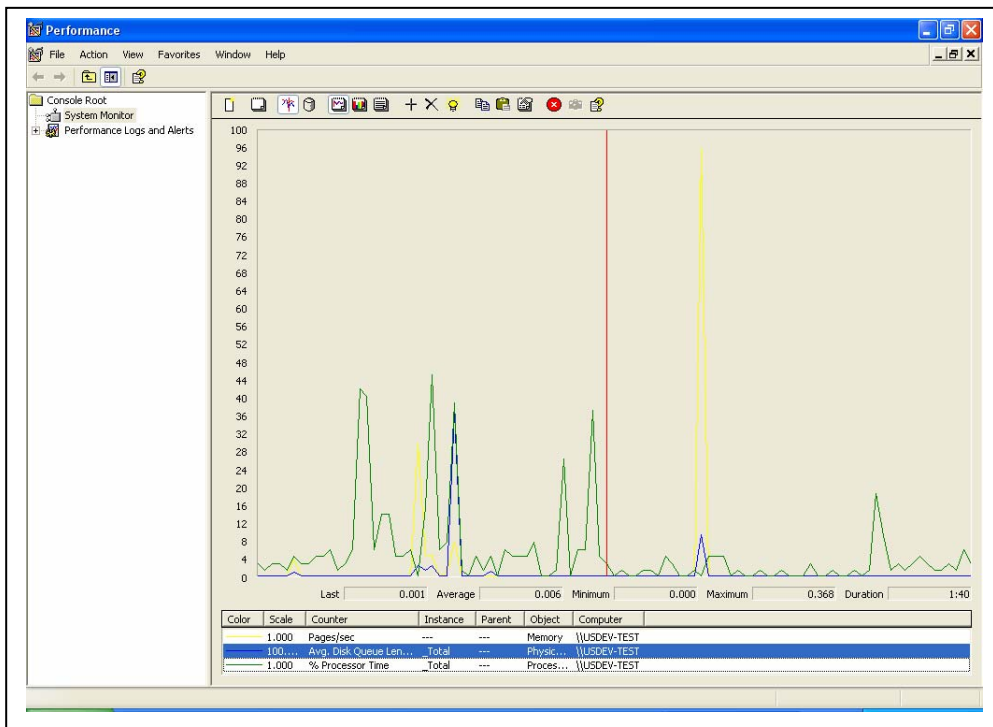
Software Components Diagram



Appendix D – Performance Monitor

Windows performance monitors will be used to monitor machine health during the stress/performance test.

Sample Performance Monitor Screen Shots



Appendix E – Trouble Tracking Form

Trouble Tracking Form

The screenshot shows a software window titled "Edit Defect" with a blue title bar and standard window controls. The form is divided into two main sections: "Primary Defect Info" and "Additional Defect Info".

Primary Defect Info:

- Name: Fix Win2k file upload 'browser bug', Change upload logic
- Description: (Empty text area)
- Project: KPG (dropdown menu)
- Date Found: 8/28/2003 (calendar icon and dropdown)
- Reported By: Ossam Manea (dropdown)
- Build Number: (Empty text field)
- Priority: High (dropdown)
- Severity: Critical (dropdown)

Additional Defect Info:

- Assigned To: Ossam Manea (dropdown)
- Status: Closed (dropdown)
- Fix Build Number: (Empty text field)
- Date Fixed: 9/ 4/2003 (calendar icon and dropdown)

Below the "Additional Defect Info" section are three tabs: "Replication Procedures" (selected), "Resolution", and "Notes". The "Replication Procedures" tab contains the text "Changed Current upload code".

At the bottom of the window are two buttons: "OK" and "Cancel".

Appendix F – Definitions and Evaluations

Client Machine

The originator machine of all the requests that need to be processed by the application. The Internet/Web browser will be running to collect data entered by the user community.

This will not be evaluated for performance. The client machine will be used to start the SOAP interface. The KPG testing will simulate 200 concurrent users and also will sustain the user volume up to 96000 users per day.

Webserver (WAE – Static Content)

The recipient machine of all the requests that need to be processed by the application. This is where the Web Application Environment is located that will manipulate a user's request. If the request can be served by the webserver then response is returned to the clients, otherwise the request is forwarded to the Application Server machine for processing. This webserver is where the static content of the WAE are kept to gain performance.

This will not be evaluated. The webserver will only contain the HTML pages used to start the SOAP interface stress/performance test.

ApplicationServer (WAE – Dynamic Content)

The recipient machine of all the requests that the webserver cannot handle. This is where the Web Application Environment is located that will manipulate a user's request that needs to be resolved since content is dynamic. The connection to the database and the application are also maintained here.

This will be evaluated for the stress/performance test for 200 concurrent users and also a sustained volume of users up to 96000 per day.

Database

Holds the data that are permanently kept in the application. There are different data access/rules defined within this layer. Database connections are also maintained on this machine.

This will be evaluated for the stress/performance test to validate the efficiencies of the SQL statements, the stored procedures, and optimized Oracle settings for the hardware where Oracle resides.

Network

WANs (Wide Area Networks) and LAN (Local Area Networks) that the requests and responses will be transported from client machine to server machine. A network switch is part of the network. This will not be evaluated due to variances in the production network.