

System Build 2

Test Plan

Version 1.0

System Build 2 Test Plan

Author's Signature

Your signature indicates that this document has been prepared with input from content experts and is in compliance with applicable system standards.

Written By:

_____ Dept. #100A931 Date: _____
Name

Reviewer's Signature

Your signature indicates that, as a content expert, you have reviewed this document and agree that it is accurate, complete, and contains the necessary degree of detail to accomplish the testing of the *System Build 2 system*.

Reviewed By:

_____ Dept. #100A641 Date: _____
Lead Analyst

_____ Dept. #100A887 Date: _____
Project Manager

Business Compliance Reviewer's Signature

Your signature indicates that this document describes a testing philosophy and approach that will demonstrate that the system will meet the business needs.

Reviewed By:

_____ Dept. #100A641 Date: _____
Implementation Manager

Computer Validation Reviewer's Signature

Your signature indicates that this document complies with applicable local requirements and corporate Computer Systems Quality policies and procedures related to computer system validation.

Reviewed By:

_____ Dept. #100A931 Date: _____
Validation Coordinator

System Custodian Approval

Your signature indicates that this document was prepared with your knowledge, you agree with the purpose and scope of this document, the document has been reviewed by the appropriate persons, and you acknowledge your responsibility in providing resources to ensure compliance.

Approved By:

_____ Dept. #100AH43 Date: _____
System Custodian

System Owner Approval

Your signature indicates that this document was prepared with your knowledge, you agree with the purpose and scope of this document, the document has been reviewed by the appropriate persons, and you acknowledge your responsibility in providing resources to ensure compliance.

Approved By:

_____ Dept. #100AC28 Date: _____
System Owner

Project Sponsor Approval

Your signature indicates that you agree with the purpose and scope of this validation strategy for the System system, and that appropriate personnel have reviewed the document to ensure compliance with company and departmental policies.

Approved By:

_____ Dept. #100A641 Date: _____
Project Sponsor

Revision History

Revision	Revision Date	Reason for Revision/Change Request	Revised By
1.0		Approved document	

Table of Contents

Terms/Acronyms	7
Introduction	8
Purpose	8
System Description	8
Overview	8
Scope	8
In Scope	8
Out of Scope	9
Audience	9
Reference Documents	9
Revisions	9
Background	10
Project Background	10
Testing Background	10
Test Planning & Execution	11
Overview	11
Testing Activities	12
Approach	13
RTM Development	13
Test Environment	14
Overview	14
Installation Qualification	15
Hardware Requirements	15
Software Requirements	15
General Requirements	15
Risk Management	16
Risks and Mitigation Strategies	16
Mitigation Strategy	16
Roles and Responsibilities	16
Unit Testing	17
Definition	17
Owner	17
Expectations of Owner	17
Approach	17
Data	17
Integration Testing	18
Definition	18
Owner	18
Expectations of Owner	18
Approach	18
Data	18
System Testing (End-to-End)	19
Definition	19
Owner	19
Expectations of Owner	19
Approach	19
Data	19
Acceptance Testing	20
Objectives	20
Owner	20
Expectations of Owner	20

Approach	20
Data Requirements.....	20
Regression Testing	21
Definition.....	21
Approach	21
Owner	21
Expectations of Owner	21
Data	21
Test Script Creation Process.....	22
Test Execution Process.....	22
Test Execution Order.....	23
Test Defect Tracking	24
Defect Criticality	24
High Criticality Defects	24
Low Criticality Defects.....	24
Test Completion	25
Test Summary Report	25
Appendix A: Unit Traceability Matrix	26
Appendix B: Order of Test Execution	27

Terms/Acronyms

Term	Explanation
Control-M	Multi-platform scheduling product used on UNIX platform with an interface to MVS to do cross platform scheduling
CORPSERV	Shared drive on Company network
CRD	Customer Reference Database
CSQ	Computer System Quality
CSV	Computer Systems Validation Policy
Dendrite	Software and Service provider for the Company US Affiliate Sales Force Automation System
FIT	Formal Integration Testing
Gemstone	US Demand Data Warehouse System
IMS	IMS America – a third party data provider
IIT	Informal Integration Testing
Informatica	Data Movement and Meta Data Tracking Technology
LAN	Local Area Network
MDL	Master Document List
ODS	Operational Data Store
Oracle	Database Resource Manager
PC	Personal Computer
Premier Force	Sales Force Automation System
QAR	Quality Assurance Review
RTM	Requirements Traceability Matrix
SFA	Sales Force Alignments
SME	Subject Matter Expert
TC	Test Coordinator
TL	Test Lead
TSR	Test Summary Report

Introduction

Purpose

System Description

The System will provide processes and procedures required to collect, store, and deliver the required data to the Rx Quota Setting process. This effort involves developing additional data feeds into the System data warehouse for collection of alignment, product, item, and quota data. Also, a data delivery mechanism will be required to deliver the appropriate data to the Rx Quota Setting process. System will be capable of supporting the Rx quota setting process, bringing closer the completion of migrating existing functionality from the Gemstone data warehouse and its associated data mart to a new information architecture that will serve the ever-growing information needs of the Company Demand organization.

Overview

This document outlines the System Build 2 Test Plan, which includes approaches for Unit, Integration and System testing. The test plan corresponds with the Testing Strategy as directed by the Quality Team.

Testing expects to identify whether requirements outlined in the approved Requirements Definition document are met. In addition, tests are designed to ensure proper development standards are in place. The Test Plan identifies the following:

- Test scope, objectives, assumptions, risks
- Roles and responsibilities
- Sequence of test activities
- Test approach for all test levels
- Direction for developing test scripts
- Test environment
- Test data
- Content and mechanism for documenting test execution.

Scope

In Scope

This Test Plan applies to the System Build 2 system. In scope for this document are:

- Alignment data feed
- Item data feed
- Rx Quota data feed
- Extracts to Rx Quota setting process
- CRD replacement data feed
- Data feed to Data Warehouse Access Layer

Out of Scope

The following is out of scope for the System Build 2 Test Plan:

- Call/Sample/Detail data
- New Starts for Humatrope
- Institutional Alignments
- Institutional Quota
- Sales Reporting Data
- IC, DDD Sales, DDD Quota
- Dendrite Data, Marketing Interventions, Alliance and PR
- Puerto Rico data
- Extract for the 'Synogy' Performance Dashboard process
- Building of a Business Object Universe to support a Business Objects user reporting front end for the access layer
- Data Warehouse data archival strategy
- Retirement or modification of Gemstone data warehouse and associated data marts

Audience

The audience for this document includes the Business Intelligence Program Sponsor, BI Program Manager, Team Leader, Validation Coordinator, Test Coordinator, Data Analyst, Technical Lead, Project Manager, Test Lead, Validation Lead, Developer and others interested in the System Build 2 project.

Reference Documents

The following documents are referenced in this document. See the *System Master Document List (MDL)* for the actual location of each document's storage location. The MDL is located at: O:\M-R\PROJNTBK\Business Intelligence Program\DW System\Validation\MDL\System MDL.doc.

- *System Master Document List (MDL)*
- *System Requirements Definition*
- *System Requirements Traceability Matrix*
- *Corporate Computer Systems Procedure Testing, CS-TST-CPR-01*
- *US Demand IT Change Management Process*

Revisions

Revisions to this test plan will follow the pre-implementation change process identified in the *US Demand IT Change Management Process*.

Background

Project Background

The Gemstone data warehouse is one of two primary systems used extensively for decision support within the Company Demand organization. One such process that resides on the Gemstone data warehouse is Rx Quota Setting. On a quarterly basis, data is extracted from the warehouse and passed to the SFO Rx quota setting process to calculate sales quotas for Company sales representatives as part of their overall compensation and incentive package.

The System Build 2 Project intends to support the Rx quota setting process. This will take us a step closer to completing the migration of existing functionality from the Gemstone data warehouse and its associated data marts to a new information architecture that will better serve the ever-growing information needs of the Company Demand organization.

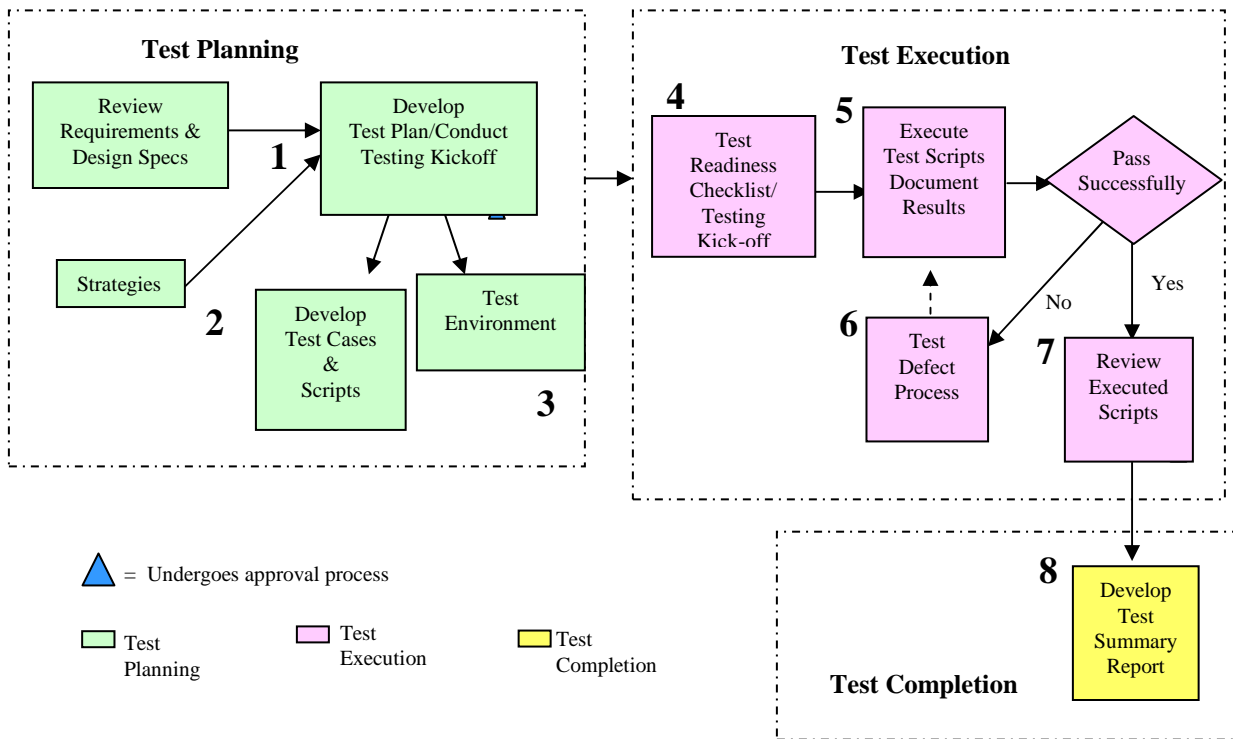
Testing Background

The purpose of testing for the System Build 2 Project is to ensure that the project satisfies the system requirements as outlined in the Requirements Definition document.

Test Planning & Execution

Overview

The following diagram illustrates the basic testing process. Each numbered item corresponds to numbered steps in the outline following the diagram.



Testing Activities

The following table summarizes the basic testing process. The numbered steps correspond to numbered items in the Test Process diagram. This process occurs for each test level: Unit, Integration and System testing.

Process	Step	Action
Test Planning Phase	1.	Develop Test Plan <ul style="list-style-type: none"> Review application requirements and/or Design Specifications, if applicable. Develop a list of in-scope deliverables for inclusion in the Test Plan. Perform risk analysis, identifying critical functions. Review and document approaches and processes for Unit, Integration, System, data conversion and parallel performance load installation, Regression, Customer Acceptance level testing and installation qualification. Document items that will be verified in Customer Acceptance testing. Complete the test plan and submit for review and approval. Identify prerequisites (training for testers, readiness of test data and testing environment) before testing can begin.
	2.	Develop Test Scripts <ul style="list-style-type: none"> Trace requirements to Test Case. Write detailed test scripts and results expected from each script. Submit test scripts for review and approval prior to formal execution
	3.	Test Environment <ul style="list-style-type: none"> Ensure that test environment is set up or verified. Ensure that all test accounts have been established and are active. Ensure installation qualification for application software program is complete.
Test Execution Phase	4.	<ul style="list-style-type: none"> Testing Kick-off Complete Test Readiness Checklist Conduct Test Kick-off Meeting
	5.	Execute Test Scripts and Capture actual Test Results <ul style="list-style-type: none"> Actual test results must be captured in its entirety and/or a screen shot that ties back to each step. A unique test run number (one that indicates how many times the test is executed) must be documented on the test script, and each iteration must be retained. (If defect uncovered, each test executed iteration must be retained) Each executed test script must be signed and dated by a reviewer, signifying that the test was properly documented and that the test results support the overall pass or fail status of the test case.
	6.	Test Defect Process <ul style="list-style-type: none"> If script fails, log test defects to document discrepancies where test results are not as expected. Depending on the outcome of the test defect, re-execute the script after the software or script has been modified. Each test script that is re-executed must be retained with a unique test run number (one that indicates how many times the test is executed).
	7.	Review Executed Test Scripts <ul style="list-style-type: none"> Ensure test results and/or documentation aligns with test steps
Test Completion Phase	8.	Develop Test Summary Report <ul style="list-style-type: none"> Document test execution results in Test Summary Report. Submit the Test Summary Report for formal approval.

Approach

A Requirement Traceability Matrix (RTM) will be developed to include the requirements within scope for this system. The document will be updated throughout the life of the system and a copy of the RTM will be included in the Test Summary Report.

RTM Development

A project's testing team owns the RTM and the related matrices that provide details for the test cases and requirements. The updates occur in the following sequence:

Step	Action	When
1.	Add each requirement to the requirement traceability matrix (RTM) in the selected tool or spreadsheet.	Upon approval of the requirements
2.	Create test cases/scripts for requirements components. Add the unique test case ID next to the appropriate requirement in the RTM.	During Unit and Integration test planning
3.	Create test cases/scripts for end-to-end testing. Add the test case/script ID next to the appropriate requirement in the matrix.	During system test planning
4.	Add the unique test run number that reflects which test iteration successfully passed	During test execution
5.	Add the status (Pass/Fail) of the test cases/scripts for each run or for the last run (manual testing).	During test execution

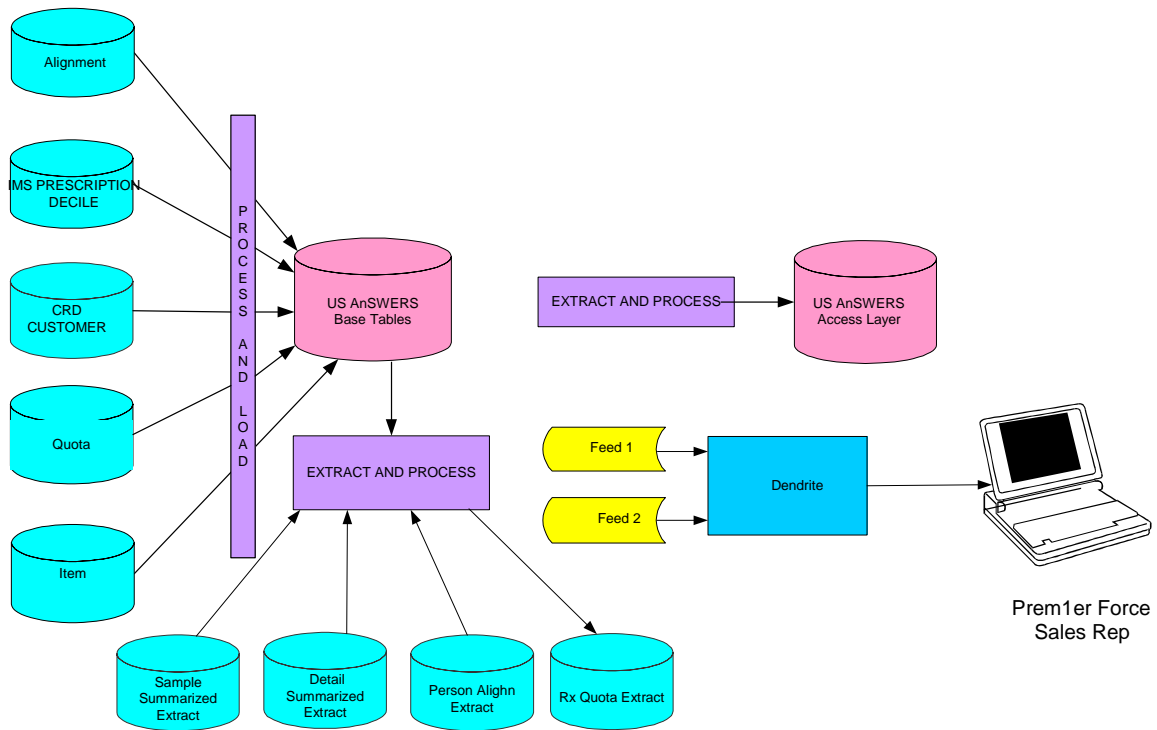
Test Environment

Overview

Informal Integration Testing will begin when development is over, at which time no development will take place. The environment is locked down for testing.

When defects are identified, access to the database by developers will be allowed. They will correct the defect and the database will be locked down again for testing.

Testing for System will consist of the following data flow:



Test Environments

Unit and Integration testing environment will consist of one Unix server (Sunrise), which is the same as that used for development. Processes will be manually run.

System test environment will consist of one Unix server, which will mirror the production environment, and Control-M will be used.

Installation Qualification

The System Build 2 project will not be using any software requiring an installation qualification.

Hardware Requirements

The following will be required:

- PC
- Laptop
- Unix Server

Software Requirements

The test environment consists of:

- Windows 2000
- Informatica PowerCenter 5
- Oracle 8i
- Premier Force 1.4
- Control-M.

General Requirements

Some general requirements must be verified before formal testing can take place:

- User logon IDs must be created and security must be established.
- Database must be loaded with the required static data necessary for the application to function.
- The test environment must be established with the necessary tools and procedures to begin testing (ability to maintain data, manipulate the system date, migrate application fixes from development, backup and restore specific problem test data, etc.).

Assumptions and Constraints

The following assumptions, constraints, and contingencies apply to this test plan.

- The person writing Test Scripts should *not* be the executor of a Test Script.
- The test data is a reasonable facsimile of the actual production environment.
- All test scripts must be approved before test execution.
- All testing processes performed by Dendrite must be completed and reviewed before approval of the final test package.
- Test Cases will be included with Test Scripts.
- The Development environment will be locked down while Unit and Integration testing are being performed.
- The QA environment will be locked down while System testing is being performed.

Risk Management

Risks and Mitigation Strategies

Potential risks to the testing efforts and the impact of the potential risks are categorized as medium, high or low. The following table shows the impact and mitigation strategies for each potential risk:

Risk	Probability of Risk (Low, Medium, High)	Impact of Risk (Low, Medium, High)	Mitigation Strategy
The executor of the test script may also be the creator due to resource constraints.	M	M	Use other resources identified in the document for the execution of the test scripts.
Test Scripts are not approved before test execution.	M	H	The TL or TC will review all scripts as they are completed prior to execution.
Development is not frozen prior to integration testing.	M	H	The development team will make the test team aware of any code changes to the testing environment and ensure no further code changes/development is in progress during test execution.
The test environment is not the same configuration/set up as the production environment.	M	H	The system testing effort will use the same configuration/set up as the production environment.

Roles and Responsibilities

See the *System Roles and Responsibilities* document for a list of roles associated with the Testing effort of this system.

Unit Testing

Definition

Unit testing is the most basic level of software testing. It has two major objectives: to verify that each component of application software works according to its specifications and to gain a high degree of confidence in the program logic.

Owner

The development team owns the technical part of unit testing activities, while the Test Lead owns the functional part of unit testing.

Expectations of Owner

The unit test owner must:

- Develop test cases and expected results
- Use Unit Traceability Matrix to record the execution of Unit Testing.
- Provide feedback regarding the RTM
- Create and identify test data
- Write and execute unit test cases
- Record test defects
- Provide information about unit testing for the TSR

Approach

The developers will perform white-box (technical) testing. Only test cases will be produced. The following functions will be tested:

- Extract alignment data from Operational Data Store (ODS) and load into System base tables
- Extract product data from Item and load into System base tables
- Extract Rx Quota data and load into System base tables
- Extract customer data from the new CRD system and load into System base tables
- Extract SAS data sets from System to RX Quota setting process
- Extract data from the System base tables and load into the System Access Layer tables

Data

All data needed for the test cases will be created via existing source files from the following sources:

- Operational Data Store (ODS) extract
- Item extract
- RX Quota extracts
- CRD extract

Integration Testing

Definition

Integration testing is testing the interface between two or more components or systems. It is ensuring that related components such as functions perform according to design specifications and that the interfaces between application software components function properly. Integration testing continues until all units are incorporated and the system is created. Integration testing includes functional testing of the interactions and the interfaces between software components and/or applications.

Owner

The Test Lead is the owner of Integration testing. The Test Lead may seek input from the developers when developing test strategy or scripts.

Expectations of Owner

The integration test owner must:

- Develop test cases/scripts and expected results
- Provide critiques of the RTM
- Generate and identify test data
- Monitors writing and execution of Integration test scripts
- Record test defects
- Provide information about System Testing for the TSR

Approach

Integration testing is performed after the completion and approval of all Unit testing. Integration testing will be performed, starting from the data sources and ending at the System Base and Access Layer tables. The processes will be executed manually so that they can be stopped at specific points to allow for validation of data. Data will be validated at the point of the System Base and Access Layer tables.

For the System Build 2 project, there will be Informal Integration Testing (IIT) and Formal Integration Testing (FIT). Informal Integration testing will occur in the Test environment. Formal Integration Testing will occur in the QA environment.

Data

Limited records will be generated to sufficiently test each test script by modifying existing source data for the following sources:

- Operational Data Store (ODS) extract
- Item extract
- RX Quota extracts
- CRD extract

System Testing (End-to-End)

Definition

System testing ensures that related components such as scripts, or functions perform according to design specifications. It is validating that the system satisfies its functional, security, and other System requirements in a simulated production environment. System testing is testing that the system runs tasks that involve more than one application or database in order to verify that the system performs the tasks accurately with the same set of data from beginning to end.

Owner

The testing team is the owner of system testing.

Expectations of Owner

The following actions are the expectations of the owner(s):

- Develop test cases and expected results for system testing.
- Provide feedback in the creation of the RTM.
- Create test data.
- Write and execute system test scripts, with a formal review before and after test execution.
- Record test defects.
- Summarize System Testing information in the Test Summary Report.

Approach

System testing is performed after the completion and approval of the Integration testing. End-to-End testing will be performed starting from data sources and ending at System Access Layer tables and the Dendrite feeds. The process will be run by Control-M.

The System team will validate the test data in System Access Layer tables and the Dendrite feeds. Processing of this data at Dendrite will not be included in the System testing since there would be no change in their test process and environment from the Integration test phase.

Data

Data for System testing will include all Integration testing records.

Acceptance Testing

Objectives

Customer Acceptance Testing (CAT) verifies that the completed system meets the original business objectives as described by the system requirements document. The client and subject matter experts are involved in planning, conducting, and reviewing this test, and the system is accepted only after successful tests.

Owner

The testing team is the owner of system testing.

Expectations of Owner

The following actions are the expectations of the owner(s):

- Develop test cases and expected results for system testing.
- Provide feedback in the creation of the RTM.
- Create test data.
- Write and execute acceptance test scripts, with a formal review before and after test execution.
- Record test defects.
- Summarize Acceptance Testing information in the Test Summary Report.

Approach

Acceptance testing takes place when all other testing is complete.

CAT will be a report to the Business experts and the Management to convey the Data Requirements

Data Requirements

Data requirement will include all System testing records.

Regression Testing

Definition

Regression testing attempts to determine whether a change to one part of an application or system adversely affects other parts. The objective of regression testing is to retest a system after modifications, maintenance or the development of a new system release.

Approach

When a test script fails, depending on the type of defect that caused the failure, it could be appropriate to execute different test scripts, which includes the new iteration.

When code changes affect the outcome of other test scripts, an analysis will be done by the developer and the test lead or test coordinator to determine the appropriate amount of regression testing.

Owner

The testing team will be the owner for most Regression testing.

Expectations of Owner

The following actions are the responsibility of the owner:

- Identify test cases/scripts and expected results for Regression testing. Create/identify test data.
- Execute Regression test scripts, with business review before and after test execution.
- Record test defects, and provide information about Regression testing for inclusion in the Test Summary Report.

Data

Data requirements will be dependant upon the stage of testing (e.g., System, Integration, etc.)

Test Script Creation Process

A test script is a detailed description of what to enter into the application and how to verify a successful or failed test. The test script can resemble a program that contains logic and variables. The test script may also resemble steps needed to execute the functionality of the application. A test case is identified within the body of the test script, a short description of a scenario, and/or path that can occur around a requirement.

During test planning, one or more test scripts will be specified for each requirement. Each test script will consist of a unique test script number (example: TS1, TS2, TS3, etc.), unique test script name, data requirements, setup requirements, purpose, assumptions/dependencies, steps to execute, and an expected result. All test scripts will be reviewed and approved prior to formal execution. After execution, the results will be reviewed.

The Testing Coordinator will conduct a Quality Assurance Review (QAR) on a subset of the test script deliverables.

Test Execution Process

Prior to execution the Test Lead will provide an approved test script package. All tests will be conducted in test UNIX\Oracle environment. If a test is executed in the UNIX\Oracle environment, it is subject to the US Demand IT Application Problem and Change Management SOP. In preparation of testing and prior to the start of test execution:

- All scripts must be in "approved" status. This status indicates test scripts have been completed and content reviewed and approved.
- The Test Lead will print and distribute the approved test script to the appropriate tester.

All testing will be conducted manually based on the step instructions within the test scripts. Each step of the test script includes the expected results of that step. All test steps are expected to be followed with no deviations. If the actual results match the expected results, the step will be marked as "passed" and the actual results will be captured in its entirety and/or a screen shot that ties back to each step.

The Tester must record the "Actual Results" of the test to assist in execution of subsequent steps or in troubleshooting subsequent defects.

- If the "expected results" has detailed information within the script, it is acceptable to write the work "passed" within the actual results field.
- If "input" data is entered by the tester, a screen shot should be printed prior to and after the execution of the test step.
- If a report is an output to a test step, the report will be printed for verification purposes. The printed screen shots must reference the step in which it is applicable. This can be hand written on the document by the tester.

Should a defect be detected, the step will be marked as "failed" on the script and a defect will be logged via the defect tracking log sheet. The status of the test script will be marked as "failed" by the tester and all execution ceases at this point. The tester will provide screen prints, when available, of the failure and attach to the test script. For more information pertaining to defect tracking, refer to the "Test Defect Tracking" section. Upon the resolution of the defect a new script must be executed and the iteration documented within the body of the script (there is a field on body of the test script for this information). Every iteration must be retained and stored in the Quality Library.

Test Execution Process, continued

When all steps in a Test Script have been successfully executed (all steps have been run and all defects have been resolved and tested or re-tested as necessary), the status of the script steps will be recorded as "passed" by the tester. At this time the Tester will attach all screen prints to the test script, sign the test script and forward it to the Test Lead, who will then route the test script to the business area for their approval and signature.

Test Execution Order

Test scripts for the various levels of testing will be executed in the following order:

- Unit Testing
- Integration Testing
- System Testing
- Regression Testing (as needed)

Test Defect Tracking

Test defects should be logged when an actual result does not match an expected result.

When a defect is discovered, the Tester will enter the defect on the defect tracking log sheet and forward it to the Test Lead along with the test script. The Test Lead logs defect to the defect-tracking log. The Test Lead assigns and forwards the defect and test script to the Developer responsible for correcting the error. Once the error has been corrected, the Developer notifies the Test Lead that the script is ready for re-execution. The Test Lead, in turn, notifies the Tester that the script is ready for re-execution. When the defect has been fixed and re-tested, the Test Lead will put the defect in a defect tracking binder. When testing activities are complete, the Test Lead records all test defects in the TSR.

Defect Criticality

Defects are classified as “high” or “low” criticality, based on how essential the functionality is to the business. The Test Lead will log the criticality of the defect. However, the criticality can change dependent upon input from the business area.

High Criticality Defects

High criticality defects are severe enough that the system cannot be implemented without fixing the problem, and there is no way to circumvent the problem.

Examples include:

- A critical screen or report does not produce expected results
- An interface does not send the necessary data and causes significant problems in the interface
- The system abnormally terminates execution on critical processes
- System data is corrupted
- Key business functions cannot be performed in a reasonable time period and there is no way to circumvent the problem
- Issues related to the product quality arise
- Issues related to health or safety arise
- Regulatory issues arise
- Defects that are highly visible to the customer and that would negatively affect customer satisfaction arise
- Translation errors that lead to incorrect interpretation of data arise
- Translations are missing

Low Criticality Defects

Low criticality defects cause the system not to work as outlined in functional and technical specifications, but does not prevent the business from performing functions. In addition, the defect does not require additional staff. Examples include:

- Misspelled words
- Formatting errors not visible to Company’s customer
- Translation errors that can be correctly interpreted by a trained user
- Defects in low criticality functions
- Errors in low criticality reports

Test Completion

Testing is complete when:

- All test scripts have been executed
- All test script results have been reviewed
- All test defects have been resolved
- TSR has been approved

Test Summary Report

The Test Summary Report is published when all testing activities are complete. The TSR includes, but is not limited to, the following:

- Introduction
- Accomplishments
- Summary of test execution for each test level
- Deviations from the Test Plan
- Unplanned events
- List of all test scripts and final statuses
- Requirements Traceability Matrix
- Test defect summary
- List of unresolved defects
- System Configuration used in testing

Appendix A: Unit Traceability Matrix

	ReqID/ Name	Test Case	Date Executed	Results	Passed /Failed	Comments, "Failed" Resolution and Retest information	Test Analyst
1							
2							
3							
4							

Appendix B: Order of Test Execution

