Pharmacy Reengineering (PRE)

**Medication Order Check Healthcare Application** **(M2.1)**

**Version 2.1**

Master Test Plan



**August 2017**

Department of Veterans Affairs

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 8/31/2017 | 2.3 | Updated the MTP section 1.3 and 3.6 | Kanika Sharma |
| 12/1/2016 | 2.2 | Updated the MTP with comments from David Skahn | Kanika Sharma |
| 10/13/2016 | 2.1 | Updated the MTP to Distinguish between MOCHA 2.1a and MOCHA 2.1b | Kanika Sharma |
| 7/28/2016 | 2.0 | Updated the comments made by reviewers (Heidi Cross and Vickey Elijah) | Kanika Sharma |
| 7/25/2016 | 1.9 | Updates made in section 1.3 and 7. Added the automation test plan to section 3 and Component Software test plan to Section 3.1 | Kanika Sharma |
| 7/30/2015 | 1.8 | Updated comments made by Vickey Elijah in section 7 | Kanika Sharma |
| 6/30/2015 | 1.7 | Updated section 1.3 and 7 with Vickey Elijah replacing Arti Iyer | Kanika Sharma |
| 4/30/2015 | 1.6 | Updated section 1.3, 7 and 8 | Kanika Sharma |
| 11/25/2014 | 1.5 | Updated the comments made by Arti and updated section 1.3 | Kanika Sharma |
| 10/27/2014 | 1.4 | Updated the comments made by Arti and updated section 1.3 | Kanika Sharma |
| 7/30/2014 | 1.3 | Updated the comments made by Arti | Kanika Sharma |
| 6/13/2014 | 1.2 | Updated the comments made by Arti | Kanika Sharma |
| 5/28/2014 | 1.1 | Updated the whole document | Kanika Sharma |
| 4/17/2013 | 1.0 | Create Initial Version | Kanika Sharma |

Table of Contents

1. Introduction 1

1.1. Purpose 1

1.2. Test Objectives 1

1.3. Roles and Responsibilities 1

1.4. Processes and References 2

2. Items To Be Tested 3

2.1. Overview of Test Inclusions 3

2.2. Overview of Test Exclusions 3

3. Test Approach 3

3.1. Product Component Test 3

3.2. Component Integration Test 3

3.3. System Tests 3

3.4. User Functionality Test 4

3.5. Enterprise System Engineering Testing 4

3.6. Initial Operating Capability Evaluation 4

4. Testing Techniques 4

4.1. Risk-based Testing 4

4.2. Enterprise Testing 4

4.2.1. Security Testing 4

4.2.2. Privacy Testing 5

4.2.3. Section 508 Compliance Testing 5

4.2.4. Multi-Divisional Testing 5

4.3. Test Types 6

4.4. Productivity and Support Tools 7

5. Test Criteria 8

5.1. Process Reviews 8

5.2. Pass/Fail Criteria 8

5.3. Suspension and Resumption Criteria 8

5.4. Acceptance Criteria 8

6. Test Deliverables 9

7. Test Schedule 9

8. Test Environments 10

8.1. Test Environment Configurations 10

8.2. Base System Hardware 10

8.3. Base Software Elements in the Test Environments 11

9. Staffing and Training Needs 12

10. Risks and Constraints 13

11. Test Metrics 13

Attachment A - Approval Signatures 13

A. Test Type Definitions 17

# Introduction

The Medication Order Check Healthcare Application (MOCHA 2.1) intends to implement Dose Range Checking with a Max Daily Dose limit for simple medication orders. The changes will be made for Outpatient Pharmacy (OP), Inpatient Medications (IP) and Pharmacy Data Management (PDM) applications. The development team will work closely with the Computerized Patient Record System (CPRS) team to make sure any corresponding changes in CPRS are also developed in this increment.

The MOCHA 2.1 increment delivers the second of four dosing increments. Dose Range Checking will be implemented using the Max Daily limit in OP, IP, PDM and CPRS for simple medication orders.

A decision was made by VA management in September 2016 to split MOCHA 2.1 into MOCHA 2.1a and MOCHA 2.1b. The reason to break out the release was:

1. To get some of the functionality out to the field quicker
2. To be able to implement Veteran – focused Integration Process (VIP) process (3 month delivery of functionality)

Below is the functionality that will be released in MOCHA 2.1a and MOCHA 2.1b

MOCHA 2.1a will provide the following enhancements:

* Add new fields to both the ADMINISTRATION SCHEDULE file (#51.1) and the MEDICATON INSTRUCTION file (#51) to define a frequency for a schedule or medication instruction used within a medication order for specific dispense drug(s) or for all drugs in order to perform a Max Daily Dose Order Check.
* Add new fields to both the ADMINISTRATION SCHEDULE file (#51.1) and the MEDICATION INSTRUCTION file (#51) to be able to derive a frequency value to perform a Max Daily Dose Order Check when the name of a schedule or medication instruction has been changed.
* Modify *Standard Schedule Edit* [PSS SCHEDULE EDIT] option to allow editing of the new frequency fields.
* Modify *Administration Schedule File Report* [PSS SCHEDULE REPORT] option to display data entered in the frequency fields.
* Modify *Medication Instruction File Add/Edit*[PSSJU MI]option to allow editing of the new frequency fields.
* Modify *Medication Instruction File Report* [PSS MED INSTRUCTION REPORT] option to display data entered in the new frequency fields.
* Modify entries to the DOSE UNITS file (#51.24), see section 4 for details.
* Create a new file called DOSE UNIT CONVERSION (#51.25) to convert one dose unit to another using a conversion factor so that a comparison can be made between two dose units when they are not equivalent.
* Add new entries to the APSP INTERVENTION TYPE file (#9009032.3), MAX DAILY DOSE and MAX SINGLE DOSE & MAX DAILY DOSE.
* Invoke CPRS Quick Order Notification when Pharmacy Orderable Item name is edited so that corresponding changes can be made to the quick order name to ensure that the Dosing Order Checks can be performed successfully.
* Enhance the free text dosage logic for dosing ranges for medication orders entered through Pharmacy and Computerized Patient Record System (CPRS).
* Enhance the free text dosage logic for multi-ingredient product medication orders entered through CPRS.
* Enhance free text logic to screen out information data placed in parenthesis which is found in the dosage ordered field for an order.

MOCHA 2.1b will provide the following enhancements:

* Implement Dose Range Checking with a Max Daily Dose limit for simple medication orders entered through Outpatient Pharmacy, Inpatient Medications applications and CPRS.
* Display an error message when the Max Daily Dose Order Check cannot be performed in CPRS, Outpatient Pharmacy, and Inpatient Medications applications.
* Apply Daily Dose Check exclusion for schedule to medication orders entered through Outpatient Pharmacy, Inpatient Medications, and CPRS.
* Apply advisory note to Max Daily Dose warning and General Dosing Guidelines for medication administered through eye, ear, or nose.
* Create a customized frequency message.
* Add First Databank (FDB) data elements from Dosing Order Check call to VistA side of interface.
* Display one warning if Maximum Single Dose and Max Daily Dose Order Check warning texts are identical.
* Exclude expired Outpatient orders from Drug Interaction Order Checks for CPRS.
* Modifications to the ‘Available Dosage(s)’ list when a screen break occurs during order entry and to the accompanying dialog during order entry through the Outpatient Pharmacy application.
* Modification to display the most recent Serum Creatinine value and date resulted if available, even if the creatinine clearance (CrCL) cannot be calculated on the pharmacy patient demographic header.
* Display body surface area (BSA) and CrCL information to the headers on all Outpatient pharmacy medication order detail screens and all Inpatient and Outpatient pharmacy patient information screen to the headers that are currently missing this information.
* Display one warning if Maximum Single Dose and Max Daily Dose Order Check warning texts are identical.

## Purpose

The purpose of this Master Test Plan is to document the overall testing approach for the MOCHA 2.1 project and to validate the MOCHA 2.1 requirements using the specified guidance in VIP.

The Test Plan defines the testing objectives, environments, roles and responsibilities, types, and methodology. The scope of testing will also be defined by identifying the functional components, the areas of high risk, and validating a representative set of data. Testing identifies the high-level business risks of the software systems involved and develops testing based on those risks.

MOCHA 2.1 implements Dose Range Checking with a Max Daily Dose limit for simple medication orders. The changes will be made for Outpatient Pharmacy, Inpatient Medications, Pharmacy Data Management (PDM) and Computerized patient record system (CPRS) applications.

## Test Objectives

This Master Test Plan supports the following objectives:

* Create a central artifact to govern the planning and control of the test effort.
* To identify, create, maintain and control the test environment.
* To provide test coverage for 100% of the documented requirements.
* To identify defects introduced by test patches and resolve them.
* Identify the motivation for and ideas behind the test areas to be covered.
* Outline the testing approach that will be used.
* List the deliverable elements of the test project.
* Identify and document the tools, techniques, standards, and methodologies to be deployed for testing.
* Identify and document the criteria for success and benchmarks to be used.

## Roles and Responsibilities

Table 1 lists the key roles and their responsibilities for this Master Test Plan.

Table : Roles and Descriptions

| Role | Description | POC |
| --- | --- | --- |
| Business Analysts | Persons who write the requirements for the defined scope of project. | Lina Bertuzis |
| Development team | Persons who build or construct the product/product component. | Ron Ruzbacki, Mai L Vo, Alberto Vargas, Hal Whitley, Chris Flegal, Asli Goncer |
| Development Manager | Person responsible for assisting with the creation and implementation of the Master Test Plan. | Heidi Cross (COR)  Scott Solden (VA PM) |
| Project Manager | Person who has overall responsibility for the successful planning and execution of the project. | Heidi Cross  Scott Solden |
| Stakeholders | Persons who hold a stake in a situation in which they may affect or be affected by the outcome. | Amy Colon, (PBM group) |
| Test Lead | An experienced Test Analyst or member of the Test team who leads and coordinates activities related to all aspects of testing based on an approved Master Test Plan and schedule. | Vickey Elijah (PRE VA SQA Lead), Kanika Sharma (SQA Contract PM)  Holly Pearson (Test Lead) |
| SQA Analyst / Test Team | Persons who execute tests and ensure the test environment will adequately support planned test activities. | Kanika Sharma, Holly Pearson, Stephen Quinn, Mehdi Balighian, Arti Sharma |
| SQA Automation Lead | Person who will be responsible to automate parts of the MOCHA application. | Farzin Navidi |
| SQA Database Patching | Person who will be responsible to patch the SQA test accounts with nationally released patches. | Mehdi Balighian, Holly Pearson, Stephen Quinn, Arti Sharma, Vickey Elijah |
| Configuration Manager | Person who establishes, maintains, and controls test environments. | Rene Kaur |

## Processes and References

The processes that guide the implementation of this Master Test Plan are:

* Test Preparation
* Product Build
* Independent Test and Evaluation

The references that support the implementation of this Master Test Plan are:

* <http://DNS/process/home.aspx>
* <http://DNS/index.asp>
* [Privacy Impact Assessment - Privacy Service](http://DNS/Privacy_Impact_Assessment.asp)

# Items To Be Tested

The requirements that will be tested for MOCHA 2.1 are Dosing (including Inpatient Medication, Outpatient Pharmacy, Pharmacy Data Management and Computerized patient record system applications), MOCHA Server 3.0, Order Check History and Event Log and they can be located at the Rational Requirements Manager (RM) link below.

[MOCHA 2.1 Requirements To Be Tested](https://DNS/rm/web#action=com.ibm.rdm.web.pages.showProjectDashboard&projectURI=https%3A%2F%2Fclm.rational.oit.va.gov%2Frm%2Fprocess%2Fproject-areas%2F_JM7DIkEeEeGP7OBhqRC-8A&vvc.configuration=https%3A%2F%2Fclm.rational.oit.va.gov%2Frm%2Fcm%2Fstream%2F_5_kBYNI6EeWADNdRzgPDK)

There are some pieces of functionality (Order Check History and Event Log) that may or may not be added as a part of the agile process. Time permitting they will be added to the product backlog and developed/tested.

## Overview of Test Inclusions

The following components and features and combinations of components and features will be tested:

* MOCHA 2.1 Dose Range Checking with Max Daily Dose Limit for simple medication orders.
* Implement Dose Range Checking with a Max Daily Dose limit for simple medication orders entered through Outpatient Pharmacy, Inpatient Medications, and CPRS applications.
* Display a generic error message when the Max Daily Dose Order Check cannot be performed in CPRS.
* Display an error message when the Max Daily Dose Order Check cannot be performed in Pharmacy with a detailed reason.
* Correct all range dose errors due to frequency failure.
* Apply Daily Dose Check exclusion for schedule to medication orders entered through Outpatient Pharmacy, Inpatient Medications and CPRS.
* Apply note to Max Daily Dose warning and General Dosing Guidelines for medication administered through eye, ear or nose.

## Overview of Test Exclusions

While this section talks about test exclusion the following components and features and combinations of components and features will be indirectly tested

These components will be tested indirectly meaning that these components are present in our test environments and as we are running tests defined for the test inclusion section we could come across an issue that will need to get corrected in the components listed below.

* COTS Drug Database - Will be tested indirectly while conducting system testing.
* Web Services – Will be tested indirectly while conducting system testing.
* XML messages – Will be tested indirectly while conducting system testing.

# Test Approach

The Test Approach section includes the tests that will be implemented for an increment; refer to the detailed test scripts/test cases in Rational Team Concert (RTC) as needed.

MOCHA 2.1 IP test cases can be found at the link below. (The links will need to be copy and pasted in a browser)

<https://DNS/qm/web/console/PHARM%20%28QM%29#action=com.ibm.rqm.planning.home.actionDispatcher&subAction=viewTestPlan&id=2106>

MOCHA 2.1 OP test cases can be found at the link below.

https://URL>DNS/qm/web/console/PHARM%20%28QM%29#action=com.ibm.rqm.planning.home.actionDispatcher&subAction=viewTestPlan&id=2107

MOCHA 2.1 PDM test cases can be found at the link below.

<https://DNS/qm/web/console/PHARM%20%28QM%29#action=com.ibm.rqm.planning.home.actionDispatcher&subAction=viewTestPlan&id=2108>

Once the developers send the build to software quality assurance (SQA) for testing it will be installed in the required test environments, pointed to a specific MOCHA Server instance and tested following the test scripts/test cases. The results will be recorded and stored in share point. If any Bugs/Defects are found they will be reported to the developers by creating defects in Rational Change and Configuration Management.

**Automation Test Plan**

The SQA Team recognizes that successful creation, maintenance, and execution of automated test scripts are the result of a flow of interdependent stages within the automation process. These stages are comprised of analysis; breaking down of scripts into distinct executable steps and identification of data that produce expected results. SQA analysts have been working closely with the VA business analysts to break down each of the existing test scripts into step by step executable line items. Expected input values are identified that would trigger a specific system behavior and produce expected results. For automated test scripts to be robust and portable to different SQA accounts and environments, SQA has gone through the time consuming process of identifying exact data input values for each expected prompt and storing this data into input files that are accessed by the test scripts during test execution. This process makes the test scripts data-driven and account independent. As a result, test scripts can be run in multiple accounts while only requiring account specific information to be stored in corresponding data files for each account. This approach saves time and effort by preventing customization and maintenance of test scripts for multiple SQA accounts. The following data driven files are used during the execution of automated test scripts:

***Automated Test Cases/Scripts:*** This entails *Automation Test Plan*, *Automation Test Strategy* and *Automation Test Suite Details* related to each script. File contains script specific information such as the script name, script type (smoke test, regression, etc.), requirements cross reference, and script completion status along with detailed description of the test script. A flag can be turned on or off for each script depending on whether or not the script needs to be run for a particular test or account.

***Automated Test Data:*** Contains test script and account specific data to allow each script to be run in multiple accounts. Not all VistA accounts include identical drug, patient or clinical information. By specifying account specific information in this file, scripts have the flexibility of running in different accounts depending on where the patch is installed.

***Automated Test Environment Configuration Details:*** This includes *Automation Test Interface Specifications* and *Automation Test Log*. It contains account, environment and interface specific data. This includes account name, IP Address and user login information needed for logging into account. Other information can be stored in this file as needed such as the wait time required between each VistA prompt as some accounts may have slower response time than others due to their configuration or amount of data in the account causing prolonged query time.

Following discussions with VA PRE management, TeamSMS has initiated creation of a baseline of automated test scripts for the existing MOCHA 2.1 functionality. Even though time consuming; this effort is necessary for establishment of a baseline that can be used to validate expected MOCHA 2.1 functionality against future enhancements. Once the baseline is established, then future automation efforts will be incorporated as part of maintenance of existing application and in conjunction with new functional requirements and development efforts. The new automated scripts will in turn become the new baseline for any future requirements and validation of new functionalities. There are 66 existing MOCHA 2.1 Regression Test Cases that have been identified as candidates for automation. Out of the 66 Regression Test Cases, 17 Test Cases have been fully automated. In addition to the Regression Test Cases, 50 Smoke Test Cases have also been fully automated for a total of 67 fully automated test scripts.

## Product Component Test

The PRE MOCHA development team will develop and execute unit tests.

SQA is planning on using Soap UI to test the changes to the MOCHA Server 3.0 using HTML test scripts. We will use input commands using HTML inputs based on actual data from Vista and capture the responses from the First Data Bank. These test script are only run in the server and the input from and to Vista will not be needed. This will isolate the server from any other possible issue being caused by an outside system. All scripts shall be checked for running without error in a test account pointed to MOCHA Server 2.0. Then the same scripts shall be run with the test account pointed to MOCHA Server 3.0. These changes should not create errors during the running of the scripts. These scripts shall be run each time a new full or partial build is delivered for testing.

SQA will run the end to end test using VistA and the developers will have the logging turned on in the MOCHA server to capture the data. SQA and the developers will turn the captured data into the HTML input files which will be used for testing.  If a script fails, the script that failed and the resulting data from the Soap UI application shall be placed in a defect and sent to the developers. This will allow the developer to run the same data on their system to uncover the code causing the issue and create a resolution for the issue.

## Component Integration Test

The PRE MOCHA SQA team will conduct integration testing with the PRE Development team. Testing completion will be documented in a spreadsheet for the test version. Anomalies and defects found will be logged in Rational Change and Configuration Management, severity and priority will be discussed in the daily huddle calls. As required by the VHA Release management for IOC exit a Software Quality Assurance Patch Checklist will be completed for each version of the patch using the automated patch check list utility created by SMS SQA. The automated checklist review will be generated using Microsoft access and will be submitted to the developer if any discrepancies are found.

## System Tests

During System Test Phase, manual testing will be conducted by the PRE SQA Team to verify the software requirements have been met, and the software is functioning appropriately. VistA, CPRS and FDB which is indirectly tested, will be used to verify enhanced order checking functionality with Outpatient Pharmacy, Inpatient Medications and Pharmacy Data Management (PDM).

The PRE test cases will be developed using the PRE Software Requirements Specification (RSD) document. Test cases will be created, managed, and executed by the MOCHA 2.1 SQA Team. The test cases will be managed in Rational Quality Manager (RQM) MOCHA 2.1 SQA will conduct a smoke test prior to full entry into the System Test cycle.

CPRS application will be tested indirectly, by allowing the user to enter all necessary orders for a patient in different packages from a single application. All pending orders that appear in the Inpatient Unit Dose and IV modules as well as the Outpatient Pharmacy prescriptions are initially entered through the CPRS package which are then finished via Pharmacy backdoor

SQA will use IBM Rational Change and Configuration Management to track any issues discovered during testing. The same process flows currently used in release management will be employed. Incidents determined to be outside the PRE domain will be logged into Rational Change and Configuration Management for resolution by the appropriate team.

## User Functionality Test

The MOCHA 2.1 SQA team will engage test sites for validating this release as part of normal procedures.

User Functionality Test (UAT) is a type of Acceptance Test that involves end-users testing the functionality of the application using test data in a controlled test environment.

|  |  |
| --- | --- |
| Technique Objective: | To verify that each requirement and business rule has been implemented as documented. |
| Technique: | Functional Testing of each section of the requirements should be completed before beginning any serious testing of those requirements in combination as a user would use the software.  Test cases are written for each individual business rule and supplementary specification. These cases are designed to test the acceptable results (positive testing) and to exceed the possible limits (negative testing) of each rule.  Functional tests are typically manually executed. Functional Testing may include testing of some integration points, but integration is not the focus of this level of testing.  Functional Testing lays the groundwork for System Testing, which will be done later in the process. |
| Method/Process: | Each test scenario has multiple execution steps and verification points. If anyone verification point fails to elicit the expected response, the entire test case is said to have failed.  Each requirement or business rule may have one or many test cases associated with it. If any one test case fails, the requirement is not validated. |
| Required Tools: | Microsoft Excel, Rational ClearQuest. Full coverage will be demonstrated by traceability to requirements. |
| Success Criteria: | When all test cases associated with a specific requirement have passed, the requirement will have been validated. |
| Special Considerations: | Each tester must have an adequate knowledge of the Pharmacy application of the business rules he/she is trying to validate. |

## Enterprise System Engineering Testing

MOCHA 2.1 SQA Team will share test scripts and results with the ESE testing team as they will review the test results and provide feedback.

## Initial Operating Capability Evaluation

The MOCHA 2.1 SQA Team will engage test sites for validating MOCHA 2.1 release as part of normal procedures. The test sites will be supported through weekly test site calls.

Subject matter experts from each site run through the test scenarios provided by the SQA team to validate each verification point.  Findings are reported to the SQA team for analysis. Upon analysis and if necessary tickets are created to address any defects or issues identified.  Fixes for identified issues are then included in subsequent builds that are installed at the test sites for re-testing. This process goes through multiple iterations until test sites are satisfied with the fixes and no further issues are reported. Once the test sites complete testing and approve the enhancements and all included fixes, they will then move forward with installing the build into their Production accounts

Test Sites for MOCHA2.1b are as follows.

Tennessee Valley

Denver

Charleston

West Palm Beach

Kansas City

# Testing Techniques

The testing techniques consist of Risk-Based Testing, Enterprise Testing, Test Types, and Productivity and Support Tools.

PRE SQA will verify System Requirements in a test environment mirroring the production accounts to the best ability it can be. During the execution of the system testing Rational Clear Quest will be used to log all incidents found. It will be the responsibility of the PMs or their designees to ensure that all issues are dealt with in a timely manner.

1. High Impact Test Incident is an error or lack of functionality that:

* Jeopardizes patient or personnel safety by corrupt or incorrect data
* Has no workaround to provide similar functionality and this functionality is required to move to system, integration, or user acceptance
* Adversely affects all users or key user functionality
* Represents a significant value or loss of life, money, or time
* Is governed by Congressional mandate
* Affects Veterans Integrated Services Network (VISN) with regards to providing consistent and safe healthcare
* Is sponsored by the National Program Office
* Negatively impacts essential operating or business processing
* System shuts down and product will not operate and cannot be kept “alive”

1. Medium Impact Test Incident is an error or lack of functionality that:

* Has a reasonable workaround to maintain functionality
* Impacts a small group of users, but has workaround
* Functionality works but not to requirements, specifications, or standards and workflow is not hampered

1. Low Impact Test Incident is an error or lack of functionality that may cause operator/user inconvenience and minimally affects operational processing:

* Spelling errors
* Minor formatting errors that do not affect functionality/visibility

1. Enhancement Test Incident is something that would be “nice” to have in the integration piece but was not included in the specifications.

## Risk-based Testing

The greatest risk to the VA community is the loss of connectivity between the CPRS Graphical User Interface (GUI) and the PRE database which could put numerous patients at risk. Error Handling for system outages for this have the highest testing priority, followed by error handling for drug errors then order based errors. Information presented for these errors must be concise to lead to quick resolution of the service interruption. PRE error handling testing will be conducted at the Drug, Order and System level to minimize this risk.

In addition, all requirements which are identified as Patient Safety Issues have the highest test priority and are verified first in the test cycle. The regression testing selected to be executed is selected to cover all the functionality areas impacted by the changes.

Regression testing was completed for high risk functionality as defined by the business owner within iterations of the builds due to schedule limitations

## Enterprise Testing

The Enterprise Testing Services (ETS) organization plans to review the test results provided by the MOCHA 2.1 SQA team.

Artifacts that will be provided to the Enterprise testing team are:

* MOCHA 2.1 Test Evaluation Summary
* IP, OP, and PDM Requirements Traceability Matrices (RTM )
* Test cases and Test results if required

### Security Testing

VistA MOCHA is a Legacy Application which does not have any built-in Security features in the application. (The users of the Pharmacy application are authenticated by the Security keys of the Local VA Medical Center). VA currently does not require MUMPS code to be tested with Fortify; however MOCHA Server code is scanned by the VA developer for security issues and is required to resolve any Critical or High findings prior to product release.

### Privacy Testing

Testing of the new functionalities is done in SQA VistA accounts at Bay Pines that only include test data and are de - identified. As a result patient privacy is not jeopardized.

### Section 508 Compliance Testing

It has been determined by the MOCHA 2.1 Project Manager that Section 508 testing will be conducted during MOCHA 2.1 testing.

During product development the MOCHA 2.1 Development Team will conduct a self-evaluation of the MOCHA 2.1 application in order to detect potential 508 compliance violations.

During functionality testing the MOCHA 2.1 Development Team and SQA team will work with the Section 508 Office to ensure that compliance testing is properly performed and the application meets the requirements as specified in the “Usability Specifications ” section of the MOCHA 2.1 Requirements Specification Document

The 508 compliance group has several checklists that they will run against the MOCHA 2.1 application to ensure that the application meets the items in the checklist. The checklists are located at:

<http://DNS/SECTION508/Standards_Checklists.asp>

The MOCHA 2.1 Project manager will obtain a Conformance Validation Statement (CVS) from the Section 508 Program Office that compliance testing was performed and validated.

### Multi-Divisional Testing

The PRE SQA team will not be able to validate this release for multiple divisions as part of normal release testing..  Multi-Divisional testing will be conducted only when testing is done by an integrated site during site testing.

Kansas City is the integrated site for MOCHA 2.1a

Tennesse Valley and Denver are integrated sites for MOCHA 2.1b

## Test Types

Table : Test Types

| Test Types | Party Responsible |
| --- | --- |
| Build verification testing | PRE SQA Team |
| Component integration testing | PRE SQA Team |
| Documentation testing | PRE SQA Team |
| Error analysis testing | PRE SQA Team |
| Exploratory testing | PRE SQA Team |
| Failover testing | PRE SQA Team |
| Installation testing | PRE SQA Team |
| Integration testing | PRE SQA Team |
| Product component testing | PRE SQA Team |
| Regression test | PRE SQA Team |
| Risk based testing | PRE SQA Team |
| Smoke testing | PRE SQA Team |
| System testing | PRE SQA Team |
| Usability testing | Test Sites and End-User workgroup |
| User Functionality Testing | Test Sites with support from PRE team |

## Productivity and Support Tools

Table 3 describes the tools that will be employed to support this Master Test Plan.

Table : Tool Category or Types

| Tool Category or Type | Tool Brand Name | Vendor or In-house | Version |
| --- | --- | --- | --- |
| Test Management | Rational Quality Manager | Rational IBM | 4.0.5 |
| Defect Tracking | Rational Team Concert | Rational IBM | 4.0.4 |
| Project Management | Project, Primavera | Microsoft | 5.0 |
| MOCHA Server Testing | (Simple Object Access Protocol) SOAP UI | SmartBear Software | 5.2.1 |
| 508 Compliance Testing | Job Access With Speech (JAWS) | Freedom Scientific | 16.0.4468 |
| Automation Testing | Rational Functional Tester (RFT) | Rational IBM | 8.5 |
| Performance Testing | Reflections |  |  |

# Test Criteria

The Test Criteria consists of Process Reviews, Pass/Fail Criteria, Suspension and Resumption Criteria, and Acceptance Criteria.

## Process Reviews

The Master Test Plan under goes two reviews:

* Peer Review – upon completion of the Master Test Plan
* Formal Review – after the Development Manager approves the Master Test Plan

The Master Test Plan does serve as an input or Artifact Used for the Process Quality Gate Review for Product Build as well as for the Go No Review (Milestone) for Independent Testing.

For more information on the reviews associated with testing, see the Product Build, Test Preparation, and Independent Test and Evaluation processes.

## Pass/Fail Criteria

Incidents identified during the execution of this test plan will be evaluated to determine their severity. This impact will be recorded in the severity section of the Rational CM Defect.

| Severity | Description |
| --- | --- |
| 1 | Critical [IEEE definition: The defect results in the failure of the complete software system, of a subsystem, or of a software unit (program or module) within the system.]   * Any defect that compromises patient safety or system security, (examples of system security defects include breach of confidentiality requirements of the Privacy Act, HIPAA or Federal Tax Information guidelines); * Loss of system functionality critical to user operations with no suitable workaround.  (I.e. there is no way to achieve the expected results using the application.) * System crash or hang that prevents further testing or operation of the complete application or a section of the application. * Any defect that causes corruption of data from a result of the system (as opposed to user error). * Any defect in which inappropriate transmissions are consistently generated or appropriate transmissions of HL7 messages fail to be generated. * Loss of functionality resulting in erroneous eligibility/enrollment determinations or communications not being sent. |
| 2 | High [IEEE definition: The defect results in the failure of the complete software system, of a subsystem, or of a software unit (program or module) within the system.]   * A major defect in the functionality which does not result in corruption of data. * A major defect in the functionality resulting in a failure of all or part of the application, where the expected results can temporarily be achieved by alternate means. The customer indicates the work around is acceptable for the short term. * Any defect that does not conform to Section 508 standards * Any defect that results in inaccurate or missing requirements * Any defect that results in invalid authentication  or authentication of an invalid end user |
| 3 | Medium [IEEE definition: The defect does not result in a failure, but causes the system to produce incorrect, incomplete, or inconsistent results, or the defect impairs the systems usability.]   * Minor functionality is not working as intended and a workaround exists but is not suitable for long term use. * The inability of a valid user to access the system consistent with granted privileges * Typographical or grammatical errors in the application, including installation guides, user guides, training manuals, design documents, etc. * Any defect producing cryptic, incorrect or inappropriate error messages * Any defect that results from the use of non-standard data terminology in the application or documentation, as defined by the Department of Veterans Affairs * Cosmetic issues that are important to the integrity of the product, but do not result in data entry and or data quality problems |

All Severity 1 and 2 defects shall be addressed or negotiated prior to release. Any limitation or outstanding test incident shall have an approved contingency process (workaround) in place.

Once the defects have been fixed by the developers, they will create unit test documents and forward them to MOCHA 2.1 SQA for validation.

Each tester must have an adequate knowledge of the Pharmacy application of the business rules in order to validate the changes to the software. When testing is completed and defects have been validated by PRE SQA, any specific ClearQuest ticket related to this testing effort will be updated with the test results.

## Suspension and Resumption Criteria

Testing will cease on a test item when an application high impact test incident is logged. Testing will resume with the incident is addressed.

## Acceptance Criteria

All Severity 1 and 2 defects shall be addressed or negotiated prior to release. Any limitation or outstanding test incident shall have an approved contingency process (workaround) in place.

# Test Deliverables

Table 4 lists the test deliverables for the MOCHA 2.1 project.

Table : Test Deliverables

| Test Deliverables | Responsible Role |
| --- | --- |
| Master Test Plan | SQA team lead |
| Test Execution Risks | Project Manager |
| Test Schedule | Project Manager |
| Test Cases/Test Scripts | PRE SQA Test team |
| Test Environment | System Administrator |
| Patch Verification (SQA Checklists) –MOCHA 2.1 | PRE SQA Test team |
| Test Evaluation Summaries | PRE SQA team lead |
| Traceability Matrix | SQA analyst, PRE SQA Test Team |

# Test Schedule

**Test Schedule can be found at the following link**

<http://DNS/projects/pre/PRE%20Schedule/Forms/AllItems.aspx?RootFolder=%2Fprojects%2Fpre%2FPRE%20Schedule%2FMOCHA%20Schedules&FolderCTID=0x012000EE60491C0AC8AF479CF3BDF4C570B869&View=%7bA0BD70BE-5A49-4402-9B83-A48F00FD1DF6%7d>

MOCHA 2.1 UFT1 ended on May 8th 2015 after completing testing on MOCHA 2.1 Combined Buildv7. Development of MOCHA 2.1 will continue after ME2 is through IOC. ME2 IOC completed February 29th 2016. As Per VA management MOCHA 2.1 UFT 2 will start in September 2016.

There will be a UFT test MOCHA 2.1 Combined Buildv8 which will be a combination of MOCHA 2.1Combined Buildv7and MOCHA Enhancement2 Nationally released Patches. This build will be ready for SQA before the MOCHA 2.1 UFT2 development starts in September 2016.

Table : Testing Milestones

| Testing Milestones | Responsible Party |
| --- | --- |
| Approved Master Test Plan | PRE SQA lead |
| Approved generic test cases (high level list) | PRE SQA lead |
| Complete and stable requirements (SRS or CRs) | PRE SQA lead |
| Creating of Test Environment(s) | PRE SQA Team |
| Submit and manage request for Testing Services | Project Managers |
| Test Cases selected for release and copied into appropriate directory in Test Manager | PRE SQA Test Team |
| Completion of Patch verification | PRE SQA Test Team |
| SQA Testing conducted (execute the selected Test Cases) in Test environment(s) | PRE SQA Test Team |
| Defects identified and entered into CQ | PRE SQA Test Team |

# Test Environments

- CHEYL36 – (Gold Account)

- CHEYL72 (Linux) – End-to-end/regression testing

- CHEYL112 – MOCHA 2.1 UFT1 (Linux) – End-to-end/regression testing

-

- MARTSQA – (Loaned to SQA by the HDR group) – VMS -Remote data testing

- CLE13 - (Loaned to SQA by the CPRS group) – VMS -Remote data testing

PRE has a GOLD legacy account to test legacy code and validate whether defects found during testing are related to the current legacy code in Production.

## Test Environment Configurations

The roles responsible for configuring and maintaining the test environments are the Configuration Manager, Test Environment team, and Software Architects.

The following Test Environment Configurations needs to be provided and supported for this project:

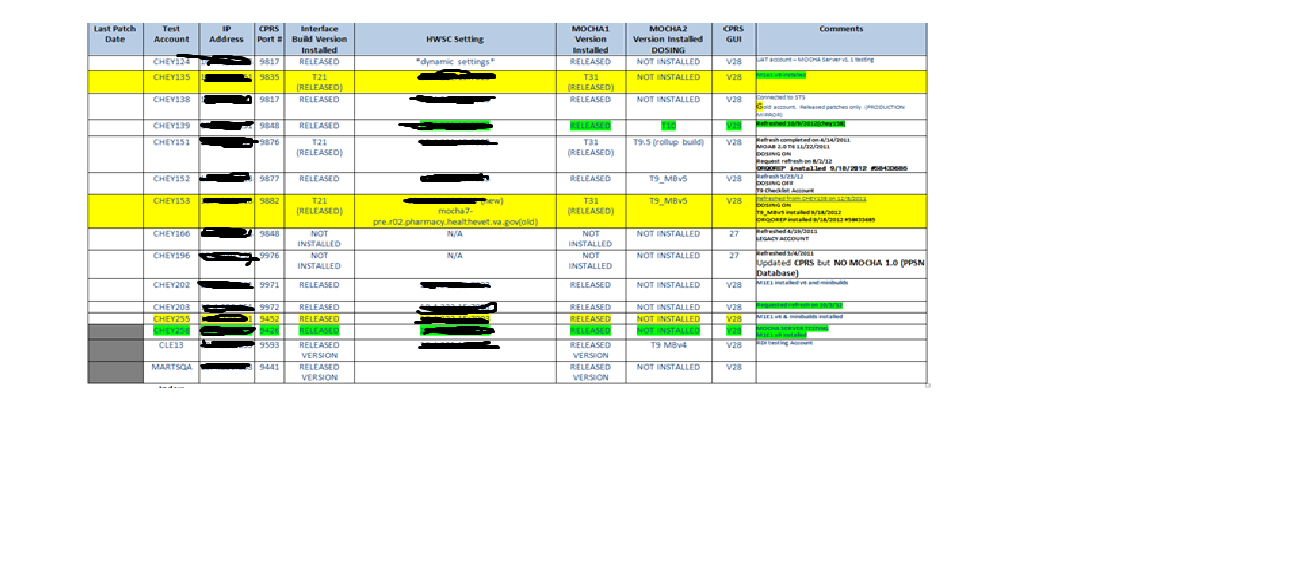
|  |  |
| --- | --- |
| Configuration Name | Description |
| Birmingham Test Center | VISTA User Class Identification (UCI)  VISTA Test and Development accounts  T1 line connection |
| Bay Pines Test Center | VISTA User Class Identification (UCI)  Web Services  XML Messaging  Message Validation Server  J2EE Servers |
| PRE Program Team | MOCHA Server  WebLogic  Java Configuration  AITC |

* MOCHA 2.1

PRE has 2 test accounts for testing the PRE changes for MOCHA 2.1. PRE also has a 1 GOLD legacy account to test legacy code and validate whether defects found during testing are related to the current legacy code in Production. PRE also share 1 account with the CPRS team to test PRE MOCHA functionality changes related to REMOTE account testing. This spreadsheet with all the accounts is kept up to date with any changes by David Savkovic and stored on the PRE Test Team SharePoint in the following link with document named “Test\_Environment\_Build\_Status\_MM\_DD\_YYYY.doc”:

<http://DNS/projects/pre/PRE_TestTeam/Testing%20Documents/Forms/AllItems.aspx?RootFolder=%2fprojects%2fpre%2fPRE%5fTestTeam%2fTesting%20Documents%2fVDD%20Components%2fTest%20Environment%20Build%20Status&FolderCTID=0x01200048706F77829BDE4AA8AAF0103A07591D>

Example of the Spread Sheet:



## Base System Hardware

Table 6 sets forth the system resources for the test effort presented in this Master Test Plan.

The specific elements of the test system may not be fully understood in early iterations, so this section may be completed over time. The test system should simulate the production environment as closely as possible, scaling down the concurrent access and database size, and so forth, if and where appropriate. Tailor the System Hardware Resources table as required.

Table : System Hardware Resources

| System Hardware Resources | | |
| --- | --- | --- |
| Resource | Quantity | Name and Type |
| Network or Subnet | 1 | VA network |
| Database Name | 1 | Cache Database |
| Client Test PCs | 6 | SQA test standard GFE machines |
| Test Repository | 1 | Share Point |
| Test Development PCs | 5 | Developer team standard GFE machines |

## 

## Base Software Elements in the Test Environments

## Table : Software Elements

| Tool Category or Type | Tool Brand Name | Vendor or In-house | Version | Use |
| --- | --- | --- | --- | --- |
| Test Management/ Testing Repository | Rational Quality Manager | IBM Rational | 4.0.5 | * Create Test cases and Suites |
| MOCHA Server3.0 | SOAP UI | SmartBear Software | 5.2.1 | * Soap UI is being used to directly test the server responses to specific data input. Outside systems are not required and eliminating the outside sources will ensure that any defect found are within the server code itself |
| 508 Testing | JAWS | Freedom Scientific | 16.0.4468 | * Test the software to make sure its accessible by all. |
| Automation Testing | RFT | IBM Rational | 8.5 | * To run the automated test scripts for MOCHA 2.1 regression testing |
| Test Incident Tracking | Rational Team Concert | IBM Rational | 4.0.5 | * Report, track, and close test incidents * Generate test reports * Tracking and closing test incidents |
| Project Management | Primavera | Primavera | 5.0 SP2 | * To record daily testing time |

# Staffing and Training Needs

Table 8 describes the personnel resources needed to plan, prepare, and execute this Master Test Plan.

Table : Staffing Resources

| Test Team | Description | Quantity | Scrum Team |
| --- | --- | --- | --- |
| Business Analysts | Persons who write the requirements for the defined scope of project. | 3 | Business Analysts |
| Software Architects | Persons who are responsible for the overall architectural design of the application | 1 | Software Architects |
| Development Team | Persons who build or construct the product/product component. | 4 | Development Team |
| Test Lead | An experienced Test Analyst or member of the Test Team who leads and coordinates activities related to all aspects of testing based on an approved Master Test Plan and schedule. | 2 | Test Lead |
| SQA Analyst / Test Team | Persons who execute tests and ensure the test environment will adequately support planned test activities. | 6 | SQA Analyst / Test Team |
| Test Environment Team | Persons who establish, maintain, and control test environments. | 2 | Test Environment Team |

# Risks and Constraints

The risks identified in this Master Test Plan may be recorded and tracked in an automated tool, such as, IBM Rational Team Concert

Risks associated with testing are potential problems/events that may cause damage to the software, system, patient, personnel, operating systems, schedule, scope, budget, and/or resources. The risks outlined here may impact scope and schedule, necessitating a deviation from this test plan. Risk impact, probability and severity classifications are defined in the Risk Management Plan.

Identified risks are entered into RTC and reviewed on Monthly Risk Management meetings.

# Test Metrics

Metrics are a system of parameters or methods for quantitative and periodic assessment of a process that is to be measured.

Test metrics may include, but are not limited to:

* Number of test cases (pass/fail)
* Percentage of test cases executed
* Number of requirements and percentage tested
* Percentage of test cases resulting in defect detection
* Number of defects attributed to test case/test script creation
* Percentage of defects identified; listed by cause and severity
* Time to re-test

# Attachment A - Approval Signatures

The Master Test Plan documents the project’s overall approach to testing and includes:

* Items to be tested
* Test strategy
* Test criteria
* Test deliverables
* Test schedule
* Test environments
* Staffing and training needs
* Risks and constraints
* Test Metrics

This section is used to document the approval of the Master Test Plan during the Formal Review.  The review should be ideally conducted face to face where signatures can be obtained ‘live’ during the review however the following forms of approval are acceptable:   
  
1.  Physical signatures obtained face to face or via fax.   
2.  Digital signatures tied cryptographically to the signer.   
3.  /es/ in the signature block provided that a separate digitally signed e-mail indicating the signer’s approval is provided and kept with the document.

NOTE: Delete the entire section above prior to final submission.

REVIEW DATE: *<date>*

*< Program/Project Manager >*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Business Sponsor Representative >*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Integrated Project Team (IPT) chair >*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Enterprise Systems Engineering (ESE) Representative >*

# A. Test Type Definitions

Test analysts use “test types” to validate the system or application under test. Simply put, test types are test techniques used to exercise the system or application. This table presents a listing of possible test types that may be utilized during the Product Build, Independent Testing, Operational Readiness Review (ORR) and Initial Operating Capability (IOC) Testing. The test analyst in consultation with the Development Manager selects the test types best suited to the system or application being tested. A minimum set of test types is suggested here. More tests may be added at the discretion of the Development Team.

|  | Product Build Testing | Independent Testing | IOC Testing |
| --- | --- | --- | --- |
| Types of Test |  |  |  |
| Access Control Testing | X |  |  |
| Benchmark Testing |  |  |  |
| Build Verification Testing | X |  |  |
| Business Cycle Testing |  |  |  |
| Compliance Testing | X |  |  |
| Component Integration Testing |  |  |  |
| Configuration Testing |  |  |  |
| Contention Testing |  |  |  |
| Data and Database Integrity Testing |  |  |  |
| Documentation Testing | X |  |  |
| Error Analysis Testing |  |  |  |
| Exploratory Testing | X |  |  |
| Failover Testing |  |  |  |
| Installation Testing | X |  |  |
| Integration Testing | X | X |  |
| Load Testing |  | X |  |
| Migration Testing |  |  |  |
| Multi-Divisional Testing |  |  |  |
| Parallel Testing |  |  |  |
| Performance Monitoring Testing |  |  |  |
| Performance Testing | X | X |  |
| Privacy Testing | X |  |  |
| Product Component Testing | X |  |  |
| Recovery Testing |  |  |  |
| Regression Test | X |  |  |
| Risk Based Testing | X |  | X |
| Section 508 Compliance Testing | X |  |  |
| Security Testing |  |  |  |
| Smoke Testing | X | X |  |
| Stress Testing |  | X |  |
| System Testing | X | X |  |
| Usability Testing | X |  | X |
| User Functionality Testing | X |  |  |
| User Interface Testing |  |  |  |

* 1. Test Type Definitions

| Test Type | Definition |
| --- | --- |
| Access Control Testing | A type of testing that attests that the target-of-test data (or systems) are accessible only to those actors for which they are intended, as defined by use cases. Access Control Testing verifies that access to the system is controlled and that unwanted or unauthorized access is prohibited. This test is implemented and executed on various targets-of-test. |
| Benchmark Testing: | A type of performance testing that compares the performance of new or unknown functionality to a known reference standard (e.g., existing software or measurements). For example, benchmark testing may compare the performance of current systems with the performance of the Linux/Oracle system. |
| Build Verification Testing  (Prerequisite: Smoke Test) | A type of testing performed for each new build, comparing the baseline with the actual object properties in the current build. The output from this test indicates what object properties have changed or don't meet the requirements. Together with the Smoke test, the Build Verification test may be utilized by projects to determine if additional functional testing is appropriate for a given build or if a build is ready for production. |
| Business Cycle Testing | A type of testing that focuses upon activities and transactions performed end to end over time. This test type executes the functionality associated with a period of time (e.g., one-week, month, or year). These tests include all daily, weekly, and monthly cycles, and events that are date-sensitive (e.g., end of the month management reports, monthly reports, quarterly reports, and year-end reports). |
| Compliance Testing | A type of testing that verifies that a collection of software and hardware fulfills given specifications. For example, these tests will minimally include: "core specifications for rehosting - ver.1.5-draft 3.doc", Section 508 of The Rehabilitation Act Amendments of 1998, Race and Ethnicity Test, and VA Directive 6102 Compliance. It does not exclude any other tests that may also come up. |
| Component Integration Testing | Testing performed to expose defects in the interfaces and interaction between integrated components as well as verifying installation instructions. |
| Configuration Testing | A type of testing concerned with checking the programs compatibility with as many possible configurations of hardware and system software. In most production environments, the particular hardware specifications for the client workstations, network connections, and database servers vary. Client workstations may have different software loaded, for example, applications, drivers, and so on hand, at any one time; many different combinations may be active using different resources. The goal of the configuration test is finding a hardware combination that should be, but is not, compatible with the program. |
| Contention Testing | A type of performance testing that executes tests that causes the application to fail with regard to actual or simulated concurrency. Contention testing identifies failures associated with locking, deadlock, livelock, starvation, race conditions, priority inversion, data loss, loss of memory, and lack of thread safety in shared software components or data. |
| Data and Database Integrity Testing | A type of testing that verifies that data is being stored by the system in a manner where the data is not compromised by the initial storage, updating, restoration, or retrieval processing. This type of testing is intended to uncover design flaws that may result in data corruption, unauthorized data access, lack of data integrity across multiple tables, and lack of adequate transaction performance. The databases, data files, and the database or data file processes should be tested as a subsystem within the application. |
| Documentation Testing | Documentation testing is a type of testing that should validate the information contained within the software documentation set for the following qualities: compliance to accepted standards and conventions, accuracy, completeness, and usability. The documentation testing should verify that all of the required information is provided in order for the appropriate user to be able to properly install, implement, operate, and maintain the software application. The current VistA documentation set can consist of any of the following manual types:  Release Notes, Installation Guide, User Manuals, Technical Manual, and Security Guide. |
| Error Analysis Testing | This type of testing verifies that the application checks for input, detects invalid data, and prevents invalid data from being entered into the application. This type of testing also includes the verification of error logs and error messages that are displayed to the user. |
| Exploratory Testing | A technique for testing computer software that requires minimal planning and tolerates limited documentation for the target-of-test in advance of test execution, relying on the skill and knowledge of the tester and feedback from test results to guide the ongoing test effort. Exploratory testing is often conducted in short sessions in which feedback gained from one session is used to dynamically plan subsequent sessions. |
| Failover Testing | A type of testing test that ensures an alternate or backup system properly "takes over" (i.e., a backup system functions when the primary system fails). Failover Testing also tests that a system continually runs when the failover occurs, and that the failover happens without any loss of data or transactions. Failover Testing should be combined with Recovery Testing. |
| Installation Testing | A type of testing that verifies that the application or system installs as intended on different hardware and software configurations, and under different conditions (e.g., a new installation, an upgrade, and a complete or custom installation). Installation testing may also measure the ease with which an application or system can be successfully installed, typically measured in terms of the average amount of person-hours required for a trained operator or hardware engineer to perform the installation. Part of this installation test is to perform an uninstall. As a result of this uninstall, the system, application and database should return to the state prior to the install. |
| Integration Testing | An incremental series of tests of combinations or sub-assemblies of selected components in an overall system. Integration testing is incremental in a successively larger and more complex combinations of components tested in sequence, proceeding from the unit level (0% integration) to eventually the full system test (100% integration). |
| Load Testing | A performance test that subjects the system to varying workloads in order to measure and evaluate the performance behaviors and abilities of the system to continue to function properly under these different workloads. Load testing determines and ensures that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics (e.g., response times, transaction rates, and other time-sensitive issues). |
| Migration Testing | A type of testing that follows standard VistA and HeV-VistA operating procedures and loads the latest .jar version onto a live copy of VistA and HeV-VistA. The following are examples of the types of tests that can be performed as part of migration testing:   * Data conversion has been completed * Data tables are successfully created * Parallel test for confirmation of data integrity * Review output report, before and after migration, to confirm data integrity * Run equivalent process, before and after migration |
| Multi-Divisional Testing | A type of testing that ensures that all applications will operate in a multi-division or multi-site environment recognizing that an enterprise perspective while fully supporting local health care delivery. |
| Parallel Testing | The same internal processes are run on the existing system and the new system. The existing system is considered the “gold standard”, unless proven otherwise. The feedback (expected results, defined time limits, data extracts, etc) from processes from the new system are compared to the existing system. Parallel testing is performed before the new system is put into a production environment. |
| Performance Monitoring Testing | Performance profiling assesses how a system is spending its time and consuming resources. This type of performance testing optimizes the performance of a system by measuring how much time and resources the system is spending in each function. These tests identify performance limitations in the code and specify which sections of the code would benefit most from optimization work. The goal of performance profiling is to optimize the feature and application performance. |
| Performance Testing | Performance Testing assesses how a system is spending its time and consuming resources. Performance testing optimizes a system by measuring how much time and resources the system is spending in each function. These tests identify performance limitations in the code and specify which sections of the code would benefit most from optimization work. Performance testing may be further refined by the use of specific types of performance tests, such as, benchmark test, load test, stress test, performance monitoring test, and contention test. |
| Privacy Testing | A type of testing that ensures that (1) veteran and employee data are adequately protected and (2) systems and applications comply with the Privacy and Security Rule provisions of the Health Insurance Portability and Accountability Act (HIPAA). |
| Product Component Testing | Product Component Testing (aka Unit Testing) is the internal technical and functional testing of a module/component of code. Product Component Testing verifies that the requirements defined in the detail design specification have been successfully applied to the module/component under test. |
| Recovery Testing | A type of testing that causes an application or system to fail in a controlled environment. Recovery processes are invoked while an application or system is monitored. Recovery testing verifies that application or system, and data recovery is achieved. Recovery Testing should be combined with Failover Testing. |
| Regression Test | A type of testing that validates existing functionality still performs as expected when new functionality is introduced into the system under test. |
| Risk Based Testing | A type of testing based on a defined list of project risks. It is designed to explore and/or uncover potential system failures by using the list of risks to select and prioritize testing. |
| Section 508 Compliance Testing | A type of test that (1) ensures that persons with disabilities have access to and are able to interact with graphical user interfaces and (2) verifies that the application or system meets the specified Section 508 Compliance standards. |
| Security Testing | A type of test that validates the security requirements and to ensure readiness for the independent testing performed by the Security Assessment Team as required by the Assessment and Authorization Process. |
| Smoke Test | A type of testing that ensures that an application or system is stable enough to enter testing in the currently active test phase. It is usually a subset of the overall set of tests, preferably automated, that touches parts of the system in at least a cursory way. |
| Stress Testing | A performance test implemented and executed to understand how a system fails due to conditions at the boundary, or outside of, the expected tolerances. This failure typically involves low resources or competition for resources. Low resource conditions reveal how the target-of-test fails that is not apparent under normal conditions. Other defects might result from competition for shared resources (e.g., database locks or network bandwidth), although some of these tests are usually addressed under functional and load testing. Stress Testing verifies the acceptability of the systems performance behavior when abnormal or extreme conditions are encountered (e.g., diminished resources or extremely high number of users). |
| System Testing | System testing is the testing of all parts of an integrated system, including interfaces to external systems. Both functional and structural types of testing are performed to verify that the system performance, operation and functionality are sound. End to end testing with all interfacing systems is the ultimate version. |
| Usability Testing | Usability testing identifies problems in the ease-of-use and ease-of-learning of a product. Usability tests may focus upon, and are not limited to: human factors, aesthetics, consistency in the user interface, online and context-sensitive help, wizards and agents, user documentation. |
| User Functionality Test | User Functionality Test (UAT) is a type of Acceptance Test that involves end-users testing the functionality of the application using test data in a controlled test environment. |
| User Interface Testing | User-interface (UI) testing exercises the user interfaces to ensure that the interfaces follow accepted standards and meet requirements. User-interface testing is often referred to as GUI testing. UI testing provides tools and services for driving the user interface of an application from a test. |