**VistA Adaptive Maintenance VAEC Security (VAM2) Master Test Plan**



**Department of Veteran Affairs (VA)**

Department of Veterans Affairs

Office of Information & Technology

Contract No: VA118-16-D-1009

task Order: 36C10B19F10090015

Deliverable: CLIN0003AA

Date: April 03, 2019

Version 1.1

Revision History

| Date | Version | Description | Author |
| --- | --- | --- | --- |
| 02-26-2019 | 1.0 | Baselined document | AbleVets |
| 04-03-2019 | 1.1 | Updated document | AbleVets |

.

Table of Contents

[1 Introduction 1](#_Toc5173649)

[1.1 Purpose of the Test Plan Document 1](#_Toc5173650)

[2 Test Objectives 1](#_Toc5173651)

[3 Test Requirements 2](#_Toc5173652)

[3.1 Data Management 2](#_Toc5173653)

[4 Test Approach 2](#_Toc5173654)

[4.1 Levels of Testing 2](#_Toc5173655)

[4.1.1 Product Component (Unit) Testing 2](#_Toc5173656)

[4.1.2 Component Integration Testing 2](#_Toc5173657)

[4.1.3 Regression Testing 2](#_Toc5173658)

[4.1.4 Smoke Testing 2](#_Toc5173659)

[4.1.5 System Testing 2](#_Toc5173660)

[4.1.6 User Functionality Testing 3](#_Toc5173661)

[4.1.7 User Acceptance Testing 3](#_Toc5173662)

[4.1.8 Security Testing 3](#_Toc5173663)

[5 Developer Level Tests 3](#_Toc5173664)

[5.1 RPC Monitor 3](#_Toc5173665)

[5.1.1 RPC Monitor Risks 3](#_Toc5173666)

[5.1.2 RPC Monitor Risks’ Mitigations 3](#_Toc5173667)

[**5.1.2.1** **We address static code quality issues as follows:** 4](#_Toc5173668)

[5.2 RPC Mirror 4](#_Toc5173670)

[5.2.1 RPC Mirror Risks 4](#_Toc5173671)

[5.2.2 RPC Mirror Risks’ Mitigations 4](#_Toc5173672)

[**5.2.2.1** **To minimize passthrough latency, our test strategy must include the following:** 4](#_Toc5173673)

[**5.2.2.2** **To mitigate the latency risks via the strategy above, we will implement the following mechanisms:** 4](#_Toc5173674)

[**5.2.2.3** **To mitigate the effects of connection overloading on the RPC Mirror our test strategy must include the following:** 5](#_Toc5173675)

[**5.2.2.4** **To mitigate the connection loading risks via the strategy above, we will implement the following mechanisms:** 5](#_Toc5173676)

[5.3 RPC Definition Models 5](#_Toc5173677)

[5.3.1 RPC Definition Models’ Risks 5](#_Toc5173678)

[5.3.2 RPC Definition Models’ Risks’ Mitigations 5](#_Toc5173679)

[6 Test Management 6](#_Toc5173682)

[7 AbleVets Delivery Pipeline: 6](#_Toc5173683)

[7.1.1 Test Deliverables 7](#_Toc5173684)

[7.1.2 Critical Testing Dependencies/Needs 7](#_Toc5173685)

List of Tables

[Table 1: Test Deliverables 6](#_Toc4671206)

# Introduction

## Purpose of the Test Plan Document

This Master Test Plan summarizes the testing approach, key objectives, test tools, and test data output for the VistA Adaptive Maintenance VAEC Security (VAM2) project. The document introduces:

* Test Strategy: The rules upon which testing will be based, and the process for establishing valid tests to output sound test data. The project’s test strategy will address the scheduling of test events, entry and exit criteria, and test data management.
* Execution Strategy: Describes how tests will be performed.
* Test Management Plan: Details test design, execution, and the process for test event issue resolution.

# Test Objectives

Testing will validate and quantify all Remote Procedure Call (RPC) volumes and coverage to generate the necessary data for resource planning, and the optimization and scaling of RPC Interface monitoring for all VA Enterprise Cloud (VAEC)-hosted VistA systems as required in Section 5.6 (Release and Deployment Support) of the Project Work Statement (PWS).

The overarching test objectives are:

* Communicate the test methods that will be employed to achieve expected results
* Leverage both manual and automated test techniques
* Perform robust testing across all solution components using VA-approved methods, tools, and reporting formats
* Conduct testing on a continuous basis (at a minimum, weekly)
* Illustrate how the test results will be validated
* Associate the deliverable test artifacts to each test performed

Specific test goals are:

* + Craft an RPC Interface test suite for all 5500 VistA RPCs and their attributes
  + Establish a set of reusable test cases for subsequent system and User Acceptance Testing (UAT)
  + Validate the thoroughness of testing, based upon the number and percentage of the RPCs audited
  + Validate the accuracy of testing, based upon RPC type, attributes, users, clients, and sensitivity
  + Provide audit and test results in a machine-processable format, to align with the use of the automated monitoring tool, CloudWatch

# Test Requirements

## Data Management

The VAM2 test team is not required to create test data, however, it will be obtained from a VistA instance in the VA environment. Once the Initial Operating Capability (IOC) test cases are defined, the VAM2 test team will determine the method for securing sensitive test data.

# Test Approach

## Levels of Testing

VAM2 will execute the following types of testing:

### Product Component (Unit) Testing

The development team will perform component testing in the local development

Environments. The developer will also execute unit testing using a tool like Jasmine. (More details in *Developer Level Tests* section)

### Component Integration Testing

The development team will perform component integration testing during Sprint testing cycles. Testing is executed to ensure that the installation and user interface is operating as designed; in addition, the interaction between the integrated components will be verified. (More details in *Developer Level Tests* section)

### Regression Testing

After one or changes/fixes have been implemented, the application is re-tested to ensure that existing functionality still performs as expected. The development and test team will perform a full regression test after defects are fixed and/or new functionalities are implemented. (More details in *Developer Level Tests* section)

### Smoke Testing

Smoke testing is the first test performed to ensure the core functionality is working and the test environment is stable enough to continue testing. The VAM2 team will perform this testing after each build is installed. (More details in *Developer Level Tests* section)

### System Testing

System testing is performed on the integrated system to evaluate compliance with the requirements; this is established by executing functional and regression testing.

### User Functionality Testing

User Functionality Testing (UFT) is executed as part of system testing; the test team will verify the front-end components and ensure the functionality of the application is working as designed*.*

### User Acceptance Testing

User Acceptance Testing (UAT) is performed during IOC by the Client to certify the system is functioning per the requirements before going into production. With guidance from the VAM2 team, end-users will develop and execute test scripts, per their testing approach, to ensure requirements are met.

### Security Testing

Security testing identifies and resolves vulnerabilities within the information system to prevent unauthorized users from accessing the system. The AbleVets Security Assessment Team will perform this test, per the A&A Process.

# Developer Level Tests

As part of the Testing and Reporting segment of the VAM2 project, we've created developer level tests and execution schemes for the following functional segments:

* RPC Monitor
* RPC Mirror
* RPC Definition Models

## RPC Monitor

The **RPC Monitor** segment represents the software pipeline that facilitates RPC parsing, classification and alert notification functions of the VAM project.

### RPC Monitor Risks

* Downtime due to runtime errors and crashes
* Degraded static code quality

### RPC Monitor Risks’ Mitigations

To address the risks of downtime due to software runtime errors, we implement the following test mechanisms for the RPC Monitor software:

* Create software unit tests using Jest. This will ensure code quality and minimize runtime errors. Unit tests shall ensure runtime coverage of at least 95%.
* Create integration-level software regression test scripts, using BATCH mode test RPC inputs.
* Every published commit to the master branch in the RPCMonitor code repository will trigger the execution of the Jest unit tests and regression tests; and will report errors and/or if the test code coverage drops below 95%. Any errors logged by the automated test execution will be fixed immediately. Running these unit tests may facilitate the need for a CI system in the VA EC, to provide access to the Kinesis mechanisms.

#### **We address static code quality issues as follows:**

* + Use a code inspection too (ESLint) and a pre-defined code formatting standard (based on the AirBnB JS standard) to ensure static code quality.
  + Every published commit to the master branch in the RPCMonitor code repository will trigger the execution of the code inspection tools, and will report broken code formatting standard rules. Any errors logged by the automated test execution will be fixed immediately.

## RPC Mirror

The **RPC Mirror** segment represents the small piece of software that mirrors client-to-VISTA RPC traffic to an alternate, data streaming service (Kinesis). This software needs to make as small an impact as possible, since it does sit on the critical network traffic path between VISTA clients and VISTA.

### RPC Mirror Risks

* App failure which would interrupt service between clients and VISTA
* Unacceptable latency introduced by the RPC mirror
* Overloading the RPC Mirror connection capacity and throughput limits

### RPC Mirror Risks’ Mitigations

To minimize the application failure rate, we implement the following test mechanisms:

* Create software unit tests using Jest. This will ensure code quality and minimize runtime errors. Unit tests shall ensure runtime coverage of at least 95%.
* Create integration-level software regression test scripts to exercise the passthrough streaming, as well as the AWS Kinesis mechanism.
* Every published commit to the master branch in the RPCMonitor code repository will trigger the execution of the Jest unit tests and regression tests, and will report errors and/or if the test code coverage drops below 95%. Any errors logged by the automated test execution will be fixed immediately. Running these unit tests may facilitate the need for a CI (ACRONYM) system in the VA EC, to provide access to the Kinesis mechanisms.

#### **To minimize passthrough latency, our test strategy must include the following:**

* + Latency characterization tests (to define acceptable latency levels and help bring them down)
  + Latency level regression tests (once acceptable levels have been determined)

#### **To mitigate the latency risks via the strategy above, we will implement the following mechanisms:**

* + Create a test harness that measures network transmission times with and without the RPC Mirror in the direct traffic path. The harness should allow for multiple, simultaneous connections and transmission streams to determine the effect of loading on the latency. The test harness should be able to run both conditions, and generate a report comparing the results from the two conditions.
  + Once we have determined nominal, acceptable latency values, the test harness will be used to run regression latency tests whenever new code is committed to the RPC Mirror repository. Any measured latency times above a certain threshold will be marked as errored and fixed immediately.

#### **To mitigate the effects of connection overloading on the RPC Mirror our test strategy must include the following:**

* + Characterization of the connection limits and capacities of a single instance of the software.
  + Load / stress regression tests (once acceptable levels have been determined)

#### **To mitigate the connection loading risks via the strategy above, we will implement the following mechanisms:**

* + Create a test harness that stress tests the RPC Mirror. The harness should be able to open an arbitrary number of raw TCP connections, then measure the response times as a function of total connection counts. This would allow us to characterize the effect of connection loading on the RPC Mirror. The harness should then be able to add enough connections to cause the single instance of the RPC Mirror to start entering failure mode. This will help us establish the upper limits of connections for a single instance of the software.
  + Once we have determined nominal and limit connection values, the test harness will be used to run regression load tests whenever new code is committed to the RPC Mirror repository. Any measured load limits below the limits characterized above will be marked as an error; and should be fixed immediately.

## RPC Definition Models

The **RPC Definition Models** segment represents the static RPC definition model files generated by the RPC Definition Toolkit, as well as the Classifier pipeline, resident in the RPC Monitor, that applies said models against RPC traffic to generate classifications and alerts.

### RPC Definition Models’ Risks

* Inconsistent classifications generated by known RPC sequences

Note: that the rest of the software operational risks should be covered by the **RPC Monitor** test segment.

### RPC Definition Models’ Risks’ Mitigations

To ensure that the RPC Definition Models and the RPC Monitor software consistently produces known results, we will do the following:

* Create RPC sequences that validate the classifier responses. This will be done by generating and running multiple sets of RPC sequences of RPCs into a test VISTA system, then verifying that the VISTA Fileman state matches the classifications generated by the classifier. In a sense, this is like a form of quasi ML training.
* Develop a validation test harness that will allow testers to run multiple known sequences of RPCs into the RPC Monitor software with a known RPC Definition model. The harness will also be able to validate the notifications and alerts generated by the RPC Monitor against a known outcome.
* Once we have determined proper function of the classifier, the test harness will be used to run regression tests whenever new code is committed to the RPC Monitor or the RPC Definition Toolkit repositories. Any deviation from the classifications set by the definition models above will be marked as an error, and should be fixed immediately.

# Test Management

The VAM2 test team is responsible for managing the testing activities throughout the testing lifecycle to include:

* Planning Phase - Creating the test plan, review and analyze requirements, allocating resources to ensure proper coverage for execution and completion of testing tasks throughout all phases, assisting in setting up the test environment, creating and updating the test schedule, and determining the functionality that will be tested
* Design Phase – Creating test data, test cases (for manual testing), and test scripts (for automated testing)
* Execution Phase – Executing test cases, documenting defects, reporting issues to developers, performing regression testing
* Closure Phase – Creating the test report and communicating results to management

# AbleVets Delivery Pipeline:

* + Code will initially be stored and tested in the public GitHub, a version control system, integrated into the AbleVets Build Pipeline and then installed in the AbleVets environment.
  + Code will be tested in the AbleVets environment and then moved to the VA environment after testing is completed:
    - Jenkins on-demand will be deployed to the test environments as well as GitHub via continuous integration of code. Based on the results, Jenkins will deploy the updated artifacts to GitHub, which will then be deployed to the Production environment.
    - VA UAT testers will develop test scripts:
      * If needed, test cases used during manual testing can be delivered to UAT testers; they do not perform automated testing.

### Test Deliverables

AbleVets will deliver the following test artifacts as per the schedule depicted below:

Table 1: Test Deliverables

|  |  |  |
| --- | --- | --- |
| **CLIN #** | **Test Deliverables** | **Frequency** |
| 0003AA | Master Test Plan | On the 3rd day of every month through the life of the project |
| 0003AB | RPC Interface Test Suite | On the 2nd day of every quarter through the life of the project |

### Critical Testing Dependencies/Needs

Testing is dependent on the following:

* Set-up AbleVets Delivery Pipeline with the latest VistA and CPRS instance
* Coordinate with the development team to identify Web-Client testing requirements
* Identify IOC test instance of VistA and CPRS
* Communication with IOC test team

Appendix A: Key Terms

The following table provides definitions for terms relevant to this document.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| A&A | Assessment & Authorization |
| CPRS | Computerized Patient Record System |
| Dev/INT | Development/Integration |
| GIT | Global Information Technology |
| HRG | Hawaii Resources Group |
| IOC | Initial Operating Capability |
| UAT | User Acceptance Testing |
| UFT | User Functionality Testing |
| UI | User Interface |
| VA | Veteran Affairs |
| VAM2 | Vista Adaptive Maintenance VAEC Security |
| VIP | Veteran-Focused Integration Process |
| VistA | Veteran Information System Technology Architecture |