VistA Adaptive Maintenance VAEC Security (VAM)

Master Test Plan



Department of Veterans Affairs

Office of Information & Technology

Contract No: VA118-16-D-1009

Task Order: 36C10B19F10090015

**November 2019**

**(No change from September 2019)**

Version 1.6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 09/03/2091 | 1.6 | Updated Master Test Plan | AbleVets |
| 08/02/2019 | 1.5 | Updated Master Test Plan | AbleVets |
| 07/03/2019 | 1.4 | Updated Master Test Plan | AbleVets |
| 06/03/2019 | 1.3 | Updated Master Test Plan | AbleVets |
| 05/03/2019 | 1.2 | Updated Master Test Plan | AbleVets |
| 04/03/2019 | 1.1 | Updated Master Test Plan | AbleVets |
| 02/26/2019 | 1.0 | Baselined Master Test Plan | AbleVets |

CLIN Satisfaction Statement

This document is submitted in satisfaction of CLIN 0003AA.

Table of Contents

[1. Introduction 1](#_Toc17978422)

[1.1. Purpose 1](#_Toc17978423)

[2. Test Objectives 1](#_Toc17978424)

[3. Levels of Testing 2](#_Toc17978425)

[3.1. Smoke Testing 2](#_Toc17978426)

[3.2. System Testing 2](#_Toc17978427)

[3.2.1. Functional Testing 2](#_Toc17978428)

[3.2.2. Regression Testing 2](#_Toc17978429)

[3.3. Product Component Testing 3](#_Toc17978430)

[3.4. Component Integration Testing 3](#_Toc17978431)

[3.5. User Acceptance Testing 3](#_Toc17978432)

[3.6. Security Testing 3](#_Toc17978433)

[4. Test and Test Data Management 3](#_Toc17978434)

[5. Test Dependencies 3](#_Toc17978435)

[6. Developer-Level Tests 4](#_Toc17978436)

[6.1. RPC Monitor 4](#_Toc17978437)

[6.1.1. RPC Monitor Risks 4](#_Toc17978438)

[6.1.2. RPC Monitor Risk Mitigation 4](#_Toc17978439)

[6.1.3. Static Code Quality 4](#_Toc17978440)

[6.2. RPC Mirror 4](#_Toc17978441)

[6.2.1. RPC Mirror Risks 4](#_Toc17978442)

[6.2.2. RPC Mirror Risk Mitigation 5](#_Toc17978443)

[6.2.3. Passthrough Latency Mitigation 5](#_Toc17978444)

[6.2.4. Connection Overload Mitigation 5](#_Toc17978445)

[6.3. RPC Definition Model 5](#_Toc17978446)

[6.3.1. RPC Definition Model Risks 6](#_Toc17978447)

[6.3.2. RPC Definition Model Risk Mitigation 6](#_Toc17978448)

[7. Code Delivery Pipeline 6](#_Toc17978449)

[8. Test Deliverables 6](#_Toc17978450)

[A. Appendix: Acronyms and Abbreviations 7](#_Toc17978451)

Table of Tables

[Table 1: Project VAM Test Deliverables 6](#_Toc17978452)

[Table 2: Acronyms and Abbreviations 7](#_Toc17978453)

# Introduction

The Veterans Health Information Systems and Technology Architecture (VistA) Adaptive Maintenance (VAM) System is a cloud-native Platform as a Service (PaaS), deployed entirely and exclusively within the Federal Risk and Authorization Management Program (FedRAMP), Health Insurance Portability and Accountability Act of 1996 (HIPAA)-compliant VA Enterprise Cloud (VAEC), leveraging the Amazon Web Services (AWS) commercial cloud infrastructure and services.

VAM provides comprehensive, commercial cloud-based monitoring and security for all clients, applications, and users of the VistA Remote Procedure Call (RPC) interface. VAM is operationalized and scaled for Enterprise Production use for all VistA systems migrated to the VAEC, leveraging FedRAMP High, VAEC-approved AWS Kinesis and AWS CloudWatch Logs.

VAM is a passive monitoring PaaS that mirrors VistA RPC traffic via AWS Kinesis to the AWS CloudWatch Logs, which is then interpreted by the RPC Monitor. AWS CloudWatch Logs are FedRAMP High certified and store all data in an encrypted form.

VAM is a 100% cloud-native, legacy-free, and non-invasive PaaS. VAM requires no change to any VistA system, nor to any end user client or application, allowing VAM to be safely and reliably deployed and scaled Enterprise-wide with minimal to no risk. Should VAM (RPC Mirror or Monitor) be disabled or deactivated, all RPC traffic flows between VistA and all its clients as usual, only without monitoring.

All of VAM’s functionality is contained exclusively and entirely as a PaaS within the VAEC, thus inheriting all security and compliance controls of the Federal Information Security Management Act of 2002 (FISMA) High VAEC. VAM has neither a connection with, nor does it share any information with, any organization, application, or system outside of the VAEC.

## Purpose

This Master Test Plan summarizes the testing approach, key objectives, test tools, and test data output for Project VAM. 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 RPC volumes and coverage to generate the necessary data for resource planning, and the optimization and scaling of RPC Interface monitoring for all VAEC-hosted VistA systems as required in Section 5.6 (Release and Deployment Support) of the Performance 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

# Levels of Testing

The following subsections describe the levels of testing Team AbleVets will conduct during the lifecycle of the project.

## 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. Team AbleVets will perform smoke testing after each build is installed.

## System Testing

System tests are performed on the integrated system to evaluate compliance with the requirements. Compliance is established by executing both functional and regression testing.

### Functional Testing

Functional testing is executed as part of system testing to validate the front-end components and ensure the functionality of the application is operating as designed.

### Regression Testing

The AbleVets Development and Test teams will perform a full regression test after defects are remediated and/or new functionality is implemented to ensure that existing functionality continues to perform as expected.

## Product Component Testing

The AbleVets Development team will perform component testing, also referred to as unit testing, in local development Environments, using a tool similar to Jasmine.

## Component Integration Testing

The AbleVets Development team will perform component integration testing during Sprint testing cycles to ensure that the installation and user interface are operating as designed, and to validate the interaction between integrated components.

## User Acceptance Testing

UAT is performed by the client during Initial Operating Capacity (IOC) to certify that the system is functioning per the requirements, prior to moving into Production. End users will develop and execute test scripts, per their testing approach, with assistance from Team AbleVets.

## Security Testing

Security testing identifies and resolves vulnerabilities within the system to prevent unauthorized access. The AbleVets Security Assessment team will perform all security tests per the Assessment and Authorization (A&A) process.

# Test and Test Data Management

Team AbleVets is responsible for managing test activities throughout the project lifecycle. The activities for each phase are detailed below.

**Planning Phase:** Create the test schedule, review and analyze requirements, determine the functionality to be tested, create the test plan, allocate resources, and assist in standing up the test environment.

**Design Phase:** Create the test data, the manual test cases, and the automated test scripts.

**Execution Phase:** Execute the test cases, document any defects identified through testing, report defects to the Development team, and perform regression testing once the defects have been remediated.

**Closure Phase:** Create post-test reports and communicate the results.

Team AbleVets is not required to create test data; however, test data will be obtained from a VistA within the VA environment. When the IOC test cases are defined, Team AbleVets will determine the method for securing sensitive test data.

# Test Dependencies

Successful testing is dependent upon the setup of the AbleVets delivery pipeline with the latest VistA and Computerized Patient Record System (CPRS) instance; coordinating with the Development team to identify web client testing requirements; identification of IOC test instances of VistA and CPRS; and open communication with IOC Test team.

# Developer-Level Tests

Team AbleVets has created Developer-level tests and execution schemes for the RPC Monitor, RPC Mirror, and RPC Definition Models.

## RPC Monitor

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

### RPC Monitor Risks

The risks associated with the RPC Monitor software are downtime due to runtime errors and crashes, and degraded quality of the static code.

### RPC Monitor Risk Mitigation

Mitigation of downtime caused by runtime errors is achieved using the Jest framework when creating unit and integration-level tests. Unit tests will ensure runtime coverage of at least 95%. Integration-level regression test scripts are created using Batch mode to test RPC inputs.

Each published commit to the master branch in the RPC Monitor code repository triggers the automated execution of Jest unit and regression tests. If the test code coverage drops below 95%, and/or if there are any errors identified during testing, they will be reported within the results. Any errors logged will be fixed immediately.

Running these tests may facilitate the need for a continuous integration system in VAEC, to provide access to the Kinesis mechanisms.

### Static Code Quality

Static code quality issues are addressed with the use of the ESLint inspection tool and a pre-defined code formatting standard. The formatting standard is based upon the Airbnb JavaScript (JS) standard.

Each published commit to the master branch in the RPC Monitor code repository triggers the automated execution of the code inspection tool. If the results indicate broken code formatting, it will be fixed immediately.

## RPC Mirror

The RPC Mirror represents the small piece of software that mirrors client-to-VistA RPC traffic to an alternate data streaming service (Kinesis). This software makes a small an impact, as it sits on the critical network traffic path between VistA clients and VistA.

### RPC Mirror Risks

The risks associated with the use of the RPC Mirror software are application failure that interrupts the service between the clients and VistA, latency introduced by the RPC Mirror, and overloading the connection capacity and/or throughput limits.

### RPC Mirror Risk Mitigation

Mitigating runtime errors and ensuring code quality is achieved using the Jest framework when creating unit and integration-level tests. Unit tests will ensure runtime coverage of at least 95%. Integration-level regression test scripts exercise passthrough streaming and the AWS Kinesis mechanism.

Each published commit to the master branch in the RPC Monitor code repository triggers the automated execution of Jest unit and regression tests. If the test code coverage drops below 95%, and/or if there are any errors identified during testing, they will be reported within the results, and fixed immediately.

Running these tests may facilitate the need for a continuous integration system in VAEC, to provide access to the Kinesis mechanisms.

### Passthrough Latency Mitigation

Passthrough latency characterization tests are used to define acceptable latency levels and establish a strategy to reduce unacceptable latency levels. Latency level regression tests are used once acceptable latency levels have been determined.

Mitigation of latency risks are addressed by creating a test harness that measures network transmission times with and without the RPC Mirror in the direct traffic path. The harness allows for multiple, simultaneous connection and transmission streams to determine the effect of loading on latency. A test results report is generated, comparing the outcomes both with and without the RPC Mirror in the direct traffic path.

The test harness is used to run regression latency tests whenever new code is committed to the RPC Mirror repository. Any measured latency times above the acceptable threshold are marked as errors and fixed immediately.

### Connection Overload Mitigation

Connection overload characterization tests are used to determine connection limits and capacities of a single instance of the software. Load/stress regression tests are used once acceptable levels have been determined.

Connection load risks are mitigated by creating an RPC Mirror stress test harness that opens an arbitrary number of raw TCP connections, then measures the response times as a function of total connection counts.

The results are used to typify the effects of connection loading on the RPC Mirror, and the test harness will be reconfigured to add enough connections to cause the single instance of the RPC Mirror to enter failure mode to establish the upper limit of connections.

The nominal and limit connection values are used to create and run regression load tests whenever new code is committed to the RPC Mirror repository. Any measured load limits above the acceptable threshold are marked as errors and fixed immediately.

## RPC Definition Model

The RPC Definition Model is the static RPC definition model file collection and the classifier pipeline. The files are generated by the RPC definition toolkit that, with the classifier pipeline, are resident in the RPC Monitor. The generated model files are applied against the RPC traffic to generate classifications and alerts.

### RPC Definition Model Risks

The risk associated with RPC Definition Models are inconsistent classifications generated by known RPC sequences.

### RPC Definition Model Risk Mitigation

Mitigation of inconsistent classifications is realized through the creation of RPC sequences that validate the classifier’s response and running multiple sets of RPC sequences into a test VistA system, to validate that the VistA FileMan state matches the classifier’s response.

Team AbleVets will ensure that the RPC Definition Models and the RPC Monitor software consistently produces known results. A test harness is created to validate the notifications and alerts generated by the RPC Monitor, and that allows testers to run multiple, known sequences of RPCs into the RPC Monitor software with a known RPC Definition Model.

The test harness runs regression tests whenever new code is committed to the RPC Mirror repository. Deviation from the known classifications set by the definition models are marked as errors and fixed immediately.

# Code Delivery Pipeline

Code is initially stored and tested in GitHub, a version control system. The code is then integrated into the AbleVets build pipeline and installed in the AbleVets test environment. Once the code testing in the AbleVets environment is complete, the code is moved to the VA environment.

Jenkins on Demand is deployed to the test environments and GitHub via the continuous integration of code. Jenkins deploys the updated artifacts to GitHub, which are then deployed to the Production environment.

VA User Acceptance (UA) testers will develop all manual UAT test scripts. If necessary, the test cases used during manual testing are delivered to UA testers. Automated testing is not conducted during UAT.

# Test Deliverables

Team AbleVets delivers the test artifacts listed in Table 1, in accordance with the PWS.

Table 1: Project VAM Test Deliverables

|  |  |  |
| --- | --- | --- |
| CLIN | Title | Frequency |
| 0003AA | Master Test Plan (MTP) | The 3rd day of every month throughout the life of the project |
| 0003AB | RPC Interface Test Suite | The 2nd day of every quarter throughout the life of the project |

1. Appendix: Acronyms and Abbreviations

Table 2 details the acronyms and abbreviations used throughout this document.

Table 2: Acronyms and Abbreviations

|  |  |
| --- | --- |
| Term | Definition |
| A&A | Assessment and Authorization |
| AWS | Amazon Web Services |
| CPRS | Computerized Patient Record System |
| FedRAMP | Federal Risk and Authorization Management Program |
| FISMA | Federal Information Security Management Act of 2002 |
| HIPAA | Health Insurance Portability and Accountability Act of 1996 |
| IOC | Initial Operating Capability |
| JS | JavaScript |
| MTP | Master Test Plan |
| OIT | Office of Information and Technology |
| PaaS | Platform as a Service |
| PoP | Period of Performance |
| PWS | Performance Work Statement |
| RPC | Remote Procedure Call |
| UA | User Acceptance |
| UAT | User Acceptance Testing |
| VA | Veteran Affairs |
| VAEC | VA Enterprise Cloud |
| VAM | Vista Adaptive Maintenance VAEC Security |
| VistA | Veteran Information System Technology Architecture |