

**DEPARTMENT OF VETERANS AFFAIRS**

**VistA Adaptive Maintenance**

# RTEP T4NG-0250

**Volume II - Technical**

Submitted to: Carolyn Carbone Contracting Officer

Department of Veterans Affairs Technology Acquisition Center 23 Christopher Way

Eatontown, NJ 07724

August 15, 2017

Submitted by:



222 Catoctin Circle SE Suite 200

Leesburg, VA 20175

Brad Palmer, Vice President of Sales

O: 571.206.0788 Ext 1002  M: 703.597.2561

[brad@b3groupinc.com](mailto:brad@b3groupinc.com)

*The data contained in this proposal is proprietary and intended for evaluation purposes only.*

# Table of Contents

[1.0 Understanding of the Requirements 1](#_bookmark0)

* 1. [Technical Approach 3](#_bookmark1)
  2. [Approach to Developing a service layer to emulate CPRS RPCs for select data read](#_bookmark2) [functions (RTEP B.1.1, PWS 5.2.1) 3](#_bookmark2)
  3. [Approach to Developing a service layer to emulate CPRS RPCs for Outpatient Pharmacy](#_bookmark3) [(CPOE) (RTEP B.1.2, PWS 5.2.2) 6](#_bookmark3)
  4. [Agile Development and Approach to Automated Testing (RTEP B.1.3, PWS 5.5.5) 7](#_bookmark4)
  5. [Approach to IOC Support (RTEP B.1.4) 9](#_bookmark5)
  6. [Approach to National Deployment while Retaining Full Functionality (RTEP B.1.5, PWS](#_bookmark6) [5.7) 12](#_bookmark6)

[POST-DEPLOYMENT WARRANTY SUPPORT 13](#_bookmark7)

* 1. [Estimated Level of Effort (RTEP B.1.6) 13](#_bookmark8)



**JAVA MEDICAL MEDICAL DOMAIN VISTA INTEGRATION DOMAIN WEB WEB SERVICES ADAPTER (VIA)**

**SERVICES (JMDWS) (MDWS)**

**WEB SERVICE**

**RPC CALL**

**RPC CALL**

**MUMPS ROUTINES/APIS**

**1.0 Understanding of the Requirements**

The Department of Veterans Affairs (VA) is a leader in the management and delivery of advanced healthcare. The VA’s pioneering clinical component of VISTA (“VISTA/EHR”) is the Computerized Patient Care System (VISTA/CPRS), providing a national approach to maintaining and accessing patient information which is critical to VA’s ability to improve health outcomes for Veterans and their dependents. But with components written in Legacy Mump and components connected via code-driven interfaces, the current ecosystem, which has organically matured to a robust system supporting 131 VistA instances, requires involved knowledge of the MUMPS programming language as well as ties-up a large percentage of resources for maintenance, growth, and usability issues.

The diagram below depicts the current access approach to VistA data at a high level. The thick 2- tier client server applications like CPRS and BCMA make direct RPC calls from the Delphi code. The client session maintains a persistent connection and a single session can connect to only one VistA instance at a time.

Web applications make use of several mechanisms to access VistA data. Medical Domain Web Services (MDWS) was written in C# and exposed a large number of RPCs as web services. Web applications can invoke this SOAP services over HTTP. VIA is written in Java and uses VistALink to call RPCs. Roll and scroll applications, written in MUMPS make calls to Fileman API directly.

COMPUTERIZED BARCODE

PATIENT MEDICATION RECORD SYSTEM ADMINISTRATIO

(CPRS) N (BCMA)

WEB CLIENT MOBILE

APPLICATIONS APPLICATIONS

ROLL AND ROLL AND

SCROLL SCROLL

APPLICATIONS APPLICATIONS



**RPC Listener**

**Remote Procedure Calls (RPC) MUMPS API**

**FILEMAN DATA**

**CLIENT**

**SOAP**

**Figure 1: Current Decentralized Fileman data access**

**VISTA**



The current PWS calls for:

* + 1. Segregating the non-clinical functions and identify all the RPCs that are used by CPRS.
    2. Create a centralized service layer that will contain the functionalities encapsulated in those RPCs used by CPRS.
    3. Provide a secure mechanism for communication to and from VistA by properly encrypting the RPC communication.
    4. Consolidate all the CPRS functionalities into a single service layer. Due to several class III development efforts, the list of RPCs can differ from one VistA system to another. Having one single middle layer would alleviate this uncertainty, and will make centralized deployment possible.
    5. Finally, a middle-layer written in a non-MUMPs language will allow future COTS products to interface with VistA seamlessly without any MUMPS programming.

The PWS especially identifies three functional domains of CPRS – Patient Vitals Data Entry, Patient Allergy Data Entry and Patient Problem Data Entry. Currently CPRS does not have the capability to enter Vitals or Allergy data. These data are entered through the roll’n’scroll interface.

Team B3 has extensive experience developing RPCs for VistA data access and manipulation and web services to invoke those RPCs. In our VIA contract, we have developed services for Orders Management and Scheduling domain. Our veteran MUMPS programmers have years of experience with VA’s implementation of VistA. They have thorough knowledge of the MUMPS programming language, which is imperative for the analysis of CPRS functions. As we will describe in our proposal, our team has in-depth knowledge of CPRS and VA clinical workflow.

Although VIA services were developed in Java, the basic principles and methodology will remain the same. Our team, through experience in multiple VA projects is familiar with the VA software development lifecycle per VIP management process and the usage of toolset mandated by the VA.

Team B3 is composed of B3 Group, Inc. and our partner, Technatomy. Our Corporate capabilities are detailed in **Table 1** below.

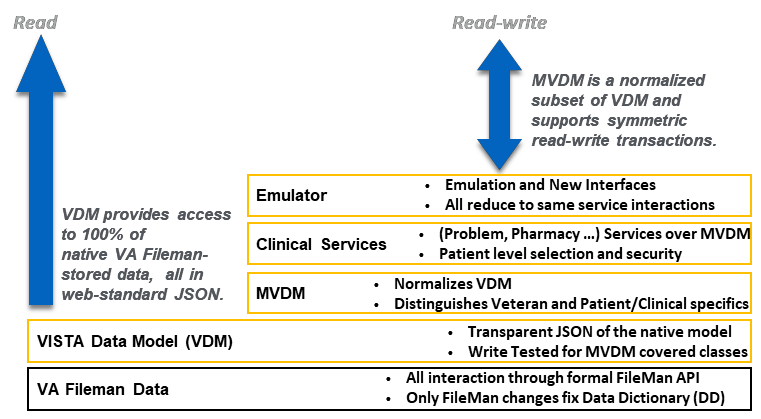
**Table 1: Team B3 Corporate Capabilities**

|  |  |
| --- | --- |
| **Company** | **Description** |
| **SDVOSB / SDB / HUBZone** | B3 Group, Inc. is a is a Center for Veterans Enterprise (CVE) verified Service-Disabled Veteran-Owned Small Business (SDVOSB), Small Disadvantaged Business, and HUBZone business headquartered in Virginia, with over nine years of support at the VA. B3 has helped the VA meet their mission of serving our nations Veterans, by providing outstanding contract support in the areas of custom development services, infrastructure support, help desk support, application support, Program Management Office (PMO) support, and technical support. |
| **Large** | Technatomy Corporation is an award-winning IT Services company built on corporate confidence and integrity. With a 250+ workforce, Technatomy implements industry- standard processes and tools based on the Project Management Institute’s (PMI) Project Management Body of Knowledge (PMBoK), our Capability Maturity Model Integration (CMMI) Maturity Level 3-assessed engineering processes, and International Organization for Standardization (ISO) quality process standards. Technatomy brings a 15-year history of providing Program Management support, Software Development, and Technical Support Services to Federal customers, including more than 10 years of service to VA. |

**2.0 Technical Approach**

## Approach to Developing a service layer to emulate CPRS RPCs for select data read functions (RTEP B.1.1, PWS 5.2.1)

Consolidating all the functionalities that are encapsulated in numerous RPCs locally deployed on 131 VistA instances will start a thorough analysis of the business logic and boundary conditions of each of the RPCs in the specified VICS. Team B3 proposes to build on the work that has already been done through the VistA data project. Following similar design patterns, we will utilize enterprise-wide Master VistA Data Model (MVDM). This data-centric, model-driven approach to VistA data allows management and interfacing through modern web standard technologies as opposed to the current MUMPS code-centric approach to interfacing VistA's data which relies on RPCs.



**Figure 2: Future Centralized Fileman Data Access**

### FileMan data modeling using web-standard technologies and representation

The core data model for VistA that serves as the roadmap for all of VA's institutional business process and clinical workflow and accompanying data - has matured over the past three decades. However, the visibility of VistA’s underlying data has always been dependent on MUMPS expertise.

MVDM abstracts VistA data with a web data model using schema-backed JSON (JSON-LD). All Fileman data is exposed and manipulated using JSON-LD. MVDM is a subset of VistA data model and is normalized across all VistA instances. This is crucial as each individual VistA can implement additional functionalities or data extensions to class 3 software. MVDM creates a “universal” data model that each VistA conforms to. MVDM provides all interfacing through a single, secure, symmetric read-write data model. Interfaces are model-driven and Fileman API compliant, and are secured with both existing Kernel authentication. Additionally, new modern, industry-standard, patient-centric, attribute-based access control (ABAC) is also implemented.

Interfaces are written using Javascript, a modern web-standard language. MVDM is symmetric, meaning the read data model is identical to the write data model. All existing clients or interfaces (such as CPRS) continue to function unchanged on top of MVDM through the RPC Emulator (a new, security isolation locker for all RPCs). All existing clients inherit all MVDM features, including enhanced patient-centric security and attribute-based access control (ABAC).

### Distinguishing VA-specific from generic healthcare patterns

VistA has been developed over thirty years to suit the specific needs of the VA. As such, the FileMan schema and data structure has been significantly modified to accommodate all business scenarios. CPRS encapsulates all the VA-specific workflows necessary for clinical delivery. Team B3’s extensive knowledge of CPRS, both from usage and programming perspective will be crucial in deciphering the business logic using numerous lessons learned from our MUMPS developers have created numerous RPCs for scheduling and orders management. We will carefully analyze the CPRS workflows related to the specified VICS to understand and document the business rules, and the RPCs invoked during the operation.

### Implementing MUMPS emulation using javascript/Node.js-driven, model-driven replacement

In order to make VistA data easily available to consuming web application, Team B3 proposes to expose VistA data for read/write capability using a JavaScript based approach. VA’s implementation of VistA is based on Intersystem’s Cache. We intend to utilize Intersystem’s add- on module for node.js environment. This add-on module can expose VISTA's data operationalized as a single, secure, symmetric read-write, server-side interface to all underlying data for external interfacing and integration.

RPCs return data delimited by “^” character. For example, querying a patient can return the following data:

^patient(6666666,"birthdate")=-2315352

^patient(6666666,"conditions",0,"causeOfDeath")=""

^patient(6666666,"conditions",0,"codes","ICD-10-CM",0)="I21.01"

^patient(6666666,"conditions",0,"end\_time")=139345842

Utilizing node.js module of Cache, the same data will be represented in JSON format as:

{

"birthdate": -2315352, "conditions": [

{

"causeOfDeath": null, "codes":

"ICD-10-CM": [ "I21.01"

]

},

"end\_time": 139345842

}

}

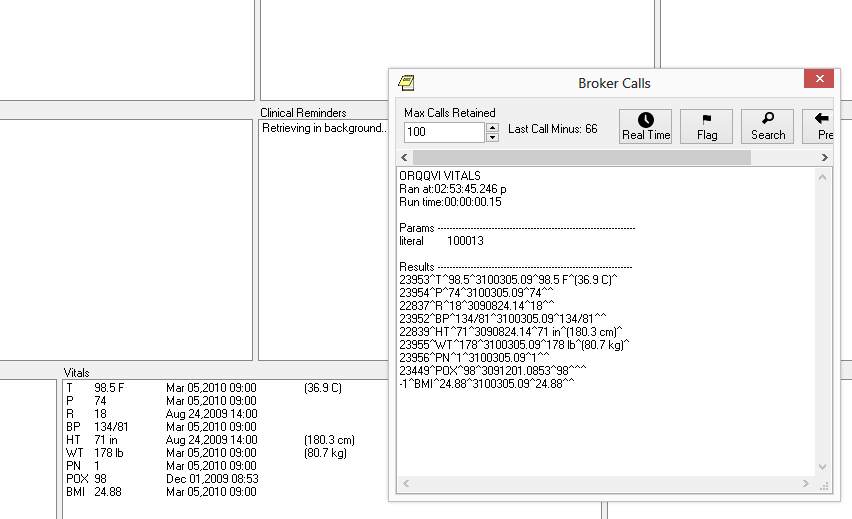
JSON is the industry standard format for REST services that can be easily consumed by modern web applications through javascripts.

### Comprehensive analysis of logic of CPRS client and its RPC interfaces

To identify the relevant CPRS RPCs that are used for specific data read functions (Vitals. Allergy and Problems), we will analyze CPRS code by running through predefined workflows. For example, Vitals and Allergy data is displayed on the CPRS cover sheet once a patient is selected. Through CPRS front end, we can determine which RPC calls are invoked.

There might be several RPCs that need to be called to accomplish a single workflow. For example, to copy inpatient medication order, seventeen RPCs are called serially, output of one acting as input to the subsequent RPC.

In case of Vitals data, we navigate through the workflow in CPRS. The following screenshot shows the RPC that fetches allergy data for a selected patient.



**Figure 3: Example of how to identify RPCs in support of CPRS**

The RPC in this scenario, ORQVI VITALS accepts one literal parameter DFN (Data File Number) for the selected patient and returns an array of result set separated by “^” delimiter.

### Operationalizing JSON models on NoSQL data stores

Modern NoSQL databases like MongoDB stores data as collections, as opposed to tables and rows in traditional RDBMS. These collections are stored in JSON format with key-value pair fields which is self-explanatory and human readable. In order to create a scheme to represent VistA data, we will analyze the MUMPS Globals and map those to collections. MUMPS itself is a NoSQL data store, however, there is no way to query the data other than MUMPS programming language.

Team B3 proposes use of MVDM ensures that all the current existing data-driven architecture of VISTA are persisted. All enhancements to any VISTA system and its data definitions will automatically be reflected in the VISTA Data Model through automated, triggered updates whenever VistA's data dictionary is updated.

Due to the organic development of the VistA RPCs, many of the MUMPS RPCs circumvent the Fileman API and writes directly to MUMPS global storage, resulting in bypassing security and auditing measures of the database. In our proposed approached using MVDM, model-view paradigm is enforced making the interfaces Model-driven, language- and client-agnostic. These interfaces will be Fileman API compliant and ensure universal auditing of all data access.

### Analysis of JLV and its VPR interface

Virtual Patient Record (VPR) extracts patient data from domains at local VistA sites to provide a cached view of a patient’s clinical chart and provides normalized fields with common field names and data structures across domains. Among others, VPR comprises two RPCs to accomplish the following:

* + 1. VPR GET PATIENT DATA: extracts VistA data in Extensible Markup Language (XML) format
    2. VPR GET PATIENT DATA JSON: extracts VistA data in JavaScript Object Notation (JSON) format

Joint Legacy Viewer (JLV) is a web-enabled GUI to view clinical information from both DoD and VA HER systems. For VA data, JLV uses VPR interface to interact with VistA. VPR RPCs are read-only, they do not write data back to VistA.

It is imperative that our proposed middleware solution implements these RPC calls so that they return identical payload in both XML and JSON format. Team B3 will analyze JLV source code and/or workflow to capture the RPCs being called by the program. Subsequently, we will utilize our web-based RPC invocation tool to test all possible scenarios with all acceptable input parameter types and compare the results returned from the services returned by new middleware.

### A final solution that has no legacy MUMPS dependencies

The goal for this task is to provide new web-based applications the ability to read from and write to VistA data without depending on MUMPS code. The service layer to emulate the RPCs need to be able to interact with VistA data directly. At the same time, a service must capture and implement all the business logic and boundary conditions encapsulated within the corresponding RPC.

Team B3 has a lab set up with FOIA VistA hosted on commercial cloud. Team B3’s experienced MUMPS developers will review existing documentation for each of the RPC that are translated. In case adequate documentation/instructions are not available, we will analyze and document each RPC to identify business logic, VistA files and fields impacted, input parameters and type and error/exception handling mechanism. We shall write test cases for each of the possible scenarios and run tests against our services. The tests will be automated and built into our Continuous Integration (CI) pipeline and will be executed after each build. Finally, the collection of node.js services will be packaged through Node Package Manager (NPM) and published for download.

## Approach to Developing a service layer to emulate CPRS RPCs for Outpatient Pharmacy (CPOE) (RTEP B.1.2, PWS 5.2.2)

Team B3’s technical approach for this task will be similar to that of the data entry functions as described in section 2.1. They only process that will be different is the analysis phase. CPOE involves providers with CPRS access and external pharmacies who fulfil the order and record the transactions through VistA roll’n’scroll interface. Additionally, MOCHA is an external service that is utilized to check drug interactions and contraindications. To analyze this process, in addition to reverse engineering CPRS for outpatient medication orders, Team B3 will perform an analysis of the external pharmacy’s workflow in VistA roll’n’scroll. We will go through each screen for pharmacy order processing and record all the business logic and file interfaces. All the data file manipulations will be implemented through node.js services as described in section 2.1.

## Agile Development and Approach to Automated Testing (RTEP B.1.3, PWS 5.5.5)

To ensure continuous software delivery, Team B3 proposes Agile methodology for performing this task order. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. Sprint planning, reviews, and demonstrations are some of the critical activities, which support these key principles. No later than two days after the completion of each sprint, the Product Owner will demonstrate to the project stakeholders and the Sprint Review Team the features developed in the sprint. The intent of the sprint review is to perform a thorough walk- through of recently developed features so that stakeholders and their representatives are able to use and assess the results independently. The application will be installed in an environment that will be readily available and accessible to remote users, allowing stakeholders and their representatives to evaluate the functionality outside of the sprint demonstration meeting. All pre- production source code will be updated and made available in the VA mandated source code repository.

Rigorous system testing is vital to developing a stable application. The Team B3 understands the pivotal nature of this task in the software development process. System testing enables us to produce a robust application (the individual components of which have been proven to work in unison). The goal of our QA team is to verify that all delivered products and documentation have met all technical and business requirements. When testing in an Agile environment, the ability to quickly react to evolving software systems and requirements is necessary. To address this need, the Team B3 will set up an automated testing environment, which will perform automated regression tests ensuring continuous system stability and functionality after software modifications. The goal is to provide complete coverage of all user stories and acceptance criteria by automation test scripts. However, when automated test scripts cannot be utilized, the Team B3 will submit written justification to both the VA Project Manager (PjM) and Contracting Officer (COR). Additionally, if a follow-up conversation or meeting is required, that will be scheduled as well. Our team is well-versed in a variety of automated testing frameworks, including SOAPUI, Selenium WebDriver RSpec (for acceptance test), JUnit (for unit test) and OpenSTA (for load testing). Upon completion of testing activities, each defect will be logged and communicated with the development team for resolution. The Team B3 will use Rational tools for reporting and tracking product’s functionality, scalability, usability, reliability, and performance related issues.

No amount of automated testing can validate or emulate the data complexity found in a real-life system like VistA. Our development and testing team has accumulated tremendous knowledge about the complexity and quirks of data, the underlying Fileman data structure and accompanying array of business rules. Also, we intend to involve VA SQA team right from the start. Our excellent

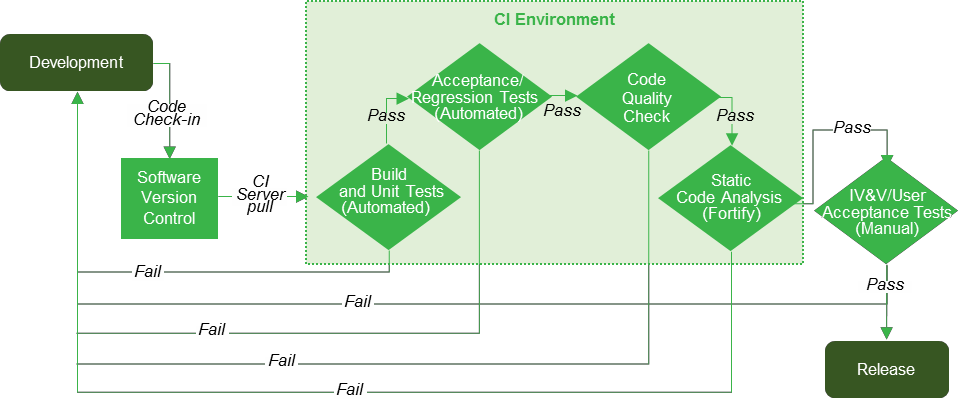
working relationship with the VA SQA personnel will ensure 100% synergy between the developers and the certifiers.

Our team’s approach is to develop a thorough Master Test Plan following the Master Test Plan Checklists, establishing a roadmap for all the QA activities that need to be performed throughout the development life cycle for each application. These deliverables will be reviewed with VA stakeholders to ensure that the integrated team all agree that they will adequately touch on potential use cases. The Master Test Plan and the associated checklist will be submitted to the COR and VA project manager for approval. Leveraging our staff’s excellent working relationship with VA, any necessary modifications to these documents will be submitted at least 15 days prior to any testing activities taking place. This will provide adequate time for the Team B3 and VA to work together to identify the changes and come to a consensus on how they should be represented within the deliverables. During each sprint, test cases will be generated based on examination of the user acceptance criteria included in each story to the extent that each transaction will be found in one or several scenarios. Our team will utilize Rational tools for developing and managing test cases.

A key part of our software testing tasks will involve reviewing, creating, updating, and editing the Master Test Plans (MTPs), test cases, test scripts, test site scenarios and acceptance criteria to reflect detailed testing coverage of the project. Test scenarios will be designed to encompass all requirements and will be stored in Rational Quality Manager (RQM). The testers will create a Requirements Traceability Matrix (RTM) to ensure that all project requirements are being met; perform negative testing to ensure that the application does not accept or allow input outside of the system's defined design; and perform regression testing to ensure that new code updates that were recently introduced do not break current functionality. Test cases and scripts will be generated in RQM so they can be easily exported to Microsoft Word or Excel for reporting purposes and included in the pre-testing package. Test scripts include inputs, objectives, detailed steps, expected and actual results, and comments. Each test case will be traced back to the RTM to ensure accuracy and thoroughness; and our testing experts will ensure that each requirement from the RSD has been covered in via the RTM.

The nature of the tasks in this PWS lends itself conveniently to fully automated testing. In our development effort, Team B3 heavily emphasizes on Test-Driven Development (TDD) approach. Unit tests are written for every function/procedure/method within a service. For node.js projects/programs, we use unit test libraries like Mocha or Chai. These unit tests are run every time the software is built. The services are tested using SoapUI, the industry standard testing tool for web services. Our experienced QA engineers have experience using SoapUI for the VIA project and we will leverage their domain knowledge of VistA and will be able to re-use these test scripts with minor modification. Team B3 has developed a command line tool to execute RPCs against any VistA system. We will execute this tool from the unit/regression tests using system call and validate the data returned by the REST service. All the test scripts are maintained in software version control system like SVN or GIT and the VA PM and other stakeholders will have real time access to the test scripts.

Team B3’s Agile development focuses on delivering working software in every iteration, and as such, its success is predicated upon frequent build and deployment. Continuous Integration/Continuous Delivery (CI/CD) plays a crucial role in this lifecycle. **Figure 4** provides an overview of our CI approach.



**Figure 4: Team B3’s Continuous Integration & Continuous Delivery Process**

1. Code-check in: Members of the development team, upon building a new feature, checks code into software version control system (GIT/SVN etc.).
2. CI process: CI server can poll the repository at regular intervals to detect any change in the codebase and start the build job. However, in our experience, this can create a bottleneck if the build job is too big and time-consuming. Depending on the nature of the build, we decide on the frequency of a particular build job.
   1. The CI Server fetches the source code from the repository and executes the compilation process. If the compile process doesn’t complete, the development team is notified. We put extra emphasis on test-driven-development, and all the unit tests are run on successful compilation.
   2. Once the automated unit tests are successfully completed, CI server triggers automated acceptance and regression tests, automated code quality/coverage check (SonarQube etc.) and Static Code Analysis (SCA) using tools like Fortify. At each of these stages, any failure is conveyed back to the development team for remediation. Fortify scan can produce “false positive” findings. Our experienced developers will review each finding and remediate the relevant findings.
   3. Once all the tests and scans are passed, we deploy the software to the Target environment (Test, SQA, Demo) for subsequent action.

Team B3 has a commercial cloud-based lab set up with three test VistA system with updated patches and fixes. We will leverage these instances for all our development and test activities in case dev/test environments are unavailable immediately. This will ensure that the team can be productive from Day 1.

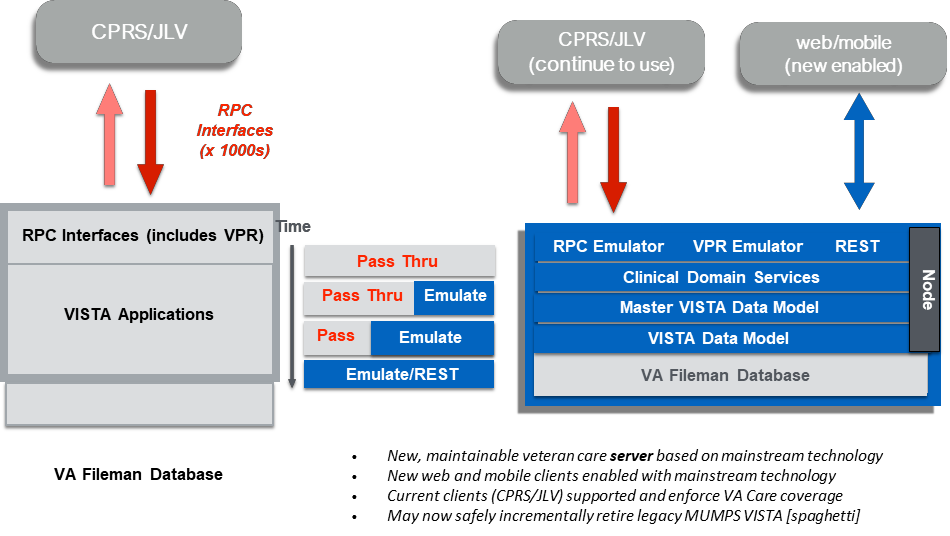
## Approach to IOC Support (RTEP B.1.4)

Following component integration/system testing (CI/ST), incremental deployment to IOC sites in support of IOC Testing using formal VA ProPath acceptance test policies and procedures and Master Test Plan. Service virtualization allows for the replication of behavior of IOC sites which allows for testing even if IOC site resources are unavailable or have restricted availability or high usage cost. This allows for IOC site specific “simulated testing” to minimize breaking current

especially since HealthShare is not able to run on virtual machines (VMs). Should a site be using a virtualized instance on VistA, will convert it to be hosted on a Linux box which not only provides new event handling and messaging capabilities but improved performance.

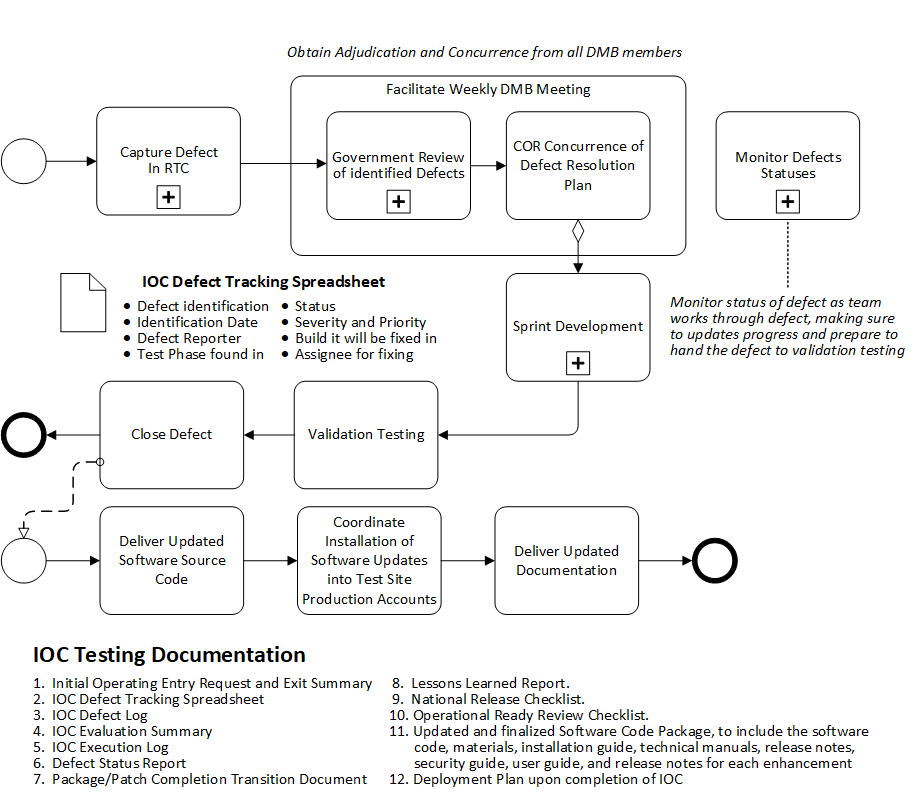
This help Team B3’s minimize disruption to CPRS users and day-to-day IOC site operations. In the seven days leading up to the go-live for each site, the deployment plan includes extensive practice of roll-back which is unlikely to occur since automated IOC deployment is done until correct. Finally, as the new functionality released, the current one does not need to be turned off since our solutions are designed to not interfere currently in production RPCs. Replacement of the legacy RPCs can be done incrementally, which begs clarification of IOC being defined as the release of the new solution intro production. The gradual replacement of the legacy service will be done during site specific downtimes via automated scripting with of course Team B3 on hand. As a result, there will be no pressure to go-live with a new system before the facility is fully prepared. Finally, a period of pre-deployment preparation and testing is conducted entirely remotely from the sites to minimize facility disruption.

In support of no impact to CPRS functionality and the sites’ day-to-day operations, Team B3 will reuse the virtualized versions of the local services that were created to support automated testing. This not only allows safe, planned, and auditable execution our Implementation Plan by being able to not only plan for but remove the need for contingencies during the IOC release of software and supporting all aspects of software deployment within the development/test platform and pre- production platform. Service virtualization allows for identification of safe automated release to IOC sites when feasible by first simulating the activities using the virtual services first. Should an error occur, a fix can be developed before release to the physical IOC site – this ability includes installing new product builds in development/test platforms, providing new product builds and tracking installation into pre-production and production platforms for IOC testing purposes. Our approach allows our IOC to readily support IOC entry effort to include patch notification messages. We will track and document the most efficient approach to install the patches.



**Figure 5: Team B3’s incremental approach to migration**

Use of SV eliminates real IOC environment defects since the release will not happen until it passes the virtual environment. However, statistically the chance of a defect does exist and Team approach to retirement of services in the local, de-centralized instances, while maintaining full continuity of service in the CPRS client provides a seamless solution to end users as we sunset the local service. Should there be an issue while transition to the centralized instance, the de- centralized is still active and functioning. Until it is accepted as migrated completely to the centralized architecture, the decentralized can serve as a “backup connection” should there be a delay in response, for example, allowing Team B3 to perform Root Cause Analysis without the end user even knowing since only the log files would show the delay. This is possible because our approach is incremental migration versus “flip the switch.” Should defects be identified, Team B3 will resolve, track and respond to all defects, as well as address all issues and questions identified during the IOC evaluation, to include security defect resolution. Remediation may include providing revised software code, repeat testing, and respond to IOC testing until IOC is successfully completed. For each defect identified, we will log the defect by establishing an IOC Defect Log, capturing the relevant severity level within Rational, and supporting the Project Manager in prioritizing the defect in the sprint backlog by participating in the DMB in order appropriately prioritize the defect, resulting in either entering it into the current sprint or the backlog within Rational as well. Using VA ProPath’s currently approved Defect Management Process, Team B3’s defect resolution identification process with provide a Defect Resolution Plan, including timeline and impacts to the schedule, software code and documentation.



**Figure 6: Team B3’s Approach to Defect Resolution**

## Approach to National Deployment while Retaining Full Functionality (RTEP B.1.5, PWS 5.7)

As a technical matter, Team B3’s approach to National Deployment while retaining full functionality will be simpler than IOC testing. National Deployment results in a single centralized service, replacing those functions of the original, 131 de-centralized VistA source instances. except simple. Just as for IOC testing, we gradually replaced the de-centralized functions, National Deployment will ensure connectivity to the data centers, local data migration, and some configuration to account for VistA instance specifics nuances – which according to VA plans should be addressed through implementation of HealthShare – but if not can be addressed through a JSON file within our solution.

Our National Deployment strategy benefits greatly from the technical simplicity of our overall technical approach. In addition, by the time National Deployment approved, a substantial test-

driven systems and software engineering capability will already be in place, providing rigorous automated testing and simulated sun setting of legacy functionalities.

VA’ s decision to host Healthcare within a VA-provided, FedRAMP approved, industry-standard, commercial cloud-based production environment. We will provide the services required to support deployment of the VistA domains and use cases of Patient Data Entry and Pharmacy Computerized Physician Order Entry (CPOE) products for installation. Team B3 will collaborate with, problem solve, trouble shoot, and provide technical support to affected VistA projects, Software Quality Assurance (SQA) and release managers.

To help coordinate releases and deliverables, Team B3 will use the POLARIS calendaring process and tool (or equivalent) will be used to track software installations, hardware replacements, system upgrades, patch release and implementation, special works in progress, and other deployment events in the VA production environment. We will provide data for populating and updating the POLARIS calendaring process for each release and deployment as well as data and information to support ATO approval as specified in the EPMO Website. We will be sure to coordinate without test site calls with VA staff to include Release Coordinators and VIP Release Agents once the product is approved for IOC production testing.

In accordance with the VIP Guide, Team B3 will provide Release and Deployment support upon successful completion of National Deployment for the first release until the warranty end of the final release. As needed, Team B3 will ensure to update all relevant documents and communicate to stakeholders:

* Update the Production Operations Manual (POM), as applicable, to include updates to regular maintenance and operations information, Responsibility, Accountability, Consulted, and Informed information, process flowcharts, dataflow diagrams, key monitoring indicators, and troubleshooting information.
* Develop the Deployment, Installation, Back-out, Rollback Plan.
* Update the User Manual, if applicable, should the adaptive maintenance impact the end users’ approach to use of the software.

### Post-Deployment Warranty Support

For a period of 90 days after the final National Deployment release is completed the Team B3 shall warrant all requirements and deliverables in the scope of this contract. Defects may be identified by the Government and its designees, to include all necessary personnel as designated by the COR. For each defect identified, the Contractor shall triage the defect, identify a resolution for the defect, and provide a Post-Deployment Warranty Support Defect Resolution Plan for resolution, including timeline and impacts to the code and updates to Rational.

**3.0 Estimated Level of Effort (RTEP B.1.6)**

Our estimated level of effort is provided below.

r e e

n i

g

En

r e e

n i

g

En

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Contract Function** | Project Manager | Node.js Developer | Node.js Developer | Data Architect | QA/Environment Manager | Cyber Security | Business Analyst, Sr | Functional/Requiremen ts Analyst | Node.js Developer | VistA MUMPS/CPRS | Junior Installation Technician/Help Desk |  |
|  | | **T4 Labor Category** | Project Manager | Developer, Sr | Developer | Architect, Sr | Test Engineer, Sr | Cyber Security | Business Analyst, Sr | Functional Analyst, Sr | Developer | Developer, Expert | Release Manager |
| **PWS** | **TITLE** | *Co\** | *B3* | *B3* | *B3* | *B3* | *B3* | *B3* | *TECH* | *TECH* | *TECH* | *TECH* | *TECH* | **Total** |
| **5.2** | **ADAPTIVE MAINTENANCE PERIOD** |  | **384** | **480** | **640** | **640** | **0** | **0** | **1920** | **960** | **480** | **384** | **0** | **5888** |
| 5.2.1 | Isolate CPRS from VistA MUMPS for Select Patient Data Entry Functions (Base Period) |  | 384 | 480 | 640 | 640 | 0 | 0 | 1920 | 960 | 480 | 384 | 0 | **5888** |
| **5.3** | **SUSTAINMENT SUPPORT FOR THE PATIENT DATA ENTRY AND PHARMACY COPE FUNCTIONS** |  | **384** | **0** | **0** | **0** | **0** | **0** | **0** | **960** | **480** | **384** | **480** | **2688** |
| **5.5** | **BUILD AND DEVELOPMENT** |  | **768** | **960** | **1280** | **1280** | **1280** | **960** | **0** | **0** | **960** | **768** | **960** | **9216** |
| 5.5.1 | Software Design |  | 128 | 160 | 214 | 214 | 214 | 100 |  |  | 160 | 128 | 160 | 1478 |
| 5.5.2 | Sprint Planning |  | 128 | 160 | 214 | 214 | 214 | 100 |  |  | 160 | 128 | 160 | 1478 |
| 5.5.3 | Sprint Execution |  | 128 | 160 | 214 | 214 | 214 | 100 |  |  | 160 | 128 | 160 | 1478 |
| 5.5.4 | System Administration and Environment Support |  | 128 | 160 | 214 | 214 | 214 | 100 |  |  | 160 | 128 | 160 | 1478 |
| 5.5.5 | Testing |  | 128 | 160 | 212 | 212 | 212 | 100 |  |  | 160 | 128 | 160 | 1472 |
| 5.5.6 | Assessment and Authorization Support (A&A) |  | 128 | 160 | 212 | 212 | 212 | 460 |  |  | 160 | 128 | 160 | 1832 |
| **5.6** | **IOC Support** |  | **384** | **480** | **0** | **0** | **640** |  | **0** | **0** | **0** | **384** | **480** | **2368** |
|  | **Base Period Total** |  | **1920** | **1920** | **1920** | **1920** | **1920** | **960** | **1920** | **1920** | **1920** | **1920** | **1920** | **20160** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5.2** | **ADAPTIVE MAINTENANCE PERIOD** |  | **192** | **480** | **640** | **640** | **0** | **0** | **1440** | **860** | **480** | **384** | **0** | **5116** |
| 5.2.1 | Isolate CPRS from VistA MUMPS for Select Patient Data Entry Functions (Base Period) |  | 192 | 480 | 640 | 640 | 0 | 0 | 1440 | 860 | 480 | 384 | 0 | **5116** |
| **5.3** | **SUSTAINMENT SUPPORT FOR THE PATIENT DATA ENTRY AND PHARMACY COPE FUNCTIONS** |  | **192** | **0** | **0** | **0** | **0** | **0** | **0** | **860** | **480** | **384** | **480** | **2396** |
| **5.5** | **BUILD AND DEVELOPMENT** |  | **1152** | **823** | **1097** | **1097** | **1097** | **810** | **0** | **0** | **823** | **624** | **823** | **8346** |
| 5.5.1 | Software Design |  | 192 | 137 | 182 | 182 | 182 | 50 |  |  | 137 | 128 | 137 | 1327 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5.5.2 | Sprint Planning |  | 192 | 138 | 183 | 183 | 183 | 50 |  |  | 138 | 128 | 138 | 1333 |
| 5.5.3 | Sprint Execution |  | 192 | 137 | 183 | 183 | 183 | 50 |  |  | 137 | 128 | 137 | 1330 |
| 5.5.4 | System Administration and Environment Support |  | 192 | 137 | 183 | 183 | 183 | 100 |  |  | 137 | 80 | 137 | 1332 |
| 5.5.5 | Testing |  | 192 | 137 | 183 | 183 | 183 | 100 |  |  | 137 | 80 | 137 | 1332 |
| 5.5.6 | Assessment and Authorization Support (A&A) |  | 192 | 137 | 183 | 183 | 183 | 460 |  |  | 137 | 80 | 137 | 1692 |
| **5.6** | **IOC Support** |  | **192** | **480** | **0** | **0** | **640** |  | **0** | **0** | **0** | **384** | **480** | **2176** |
| **5.7** | **RELEASE AND DEPLOYMENT SUPPORT** |  | **192** | **137** | **183** | **183** | **183** | **150** | **480** | **200** | **137** | **144** | **137** | **2126** |
| **5.7.1** | **Post Deployment Warranty Support** |  | **192** | **137** | **183** | **183** | **183** | **150** | **480** | **200** | **137** | **144** | **137** | **2126** |
|  | **Option Period Total** |  | **1920** | **1920** | **1920** | **1920** | **1920** | **960** | **1920** | **1920** | **1920** | **1920** | **1920** | **20160** |