

**T4NG-0250 [TAC-17-42242] Technical Proposal Volume II**

**VistA Adaptive Maintenance**

# August 17, 2017

**Submitted to:**

**Department of Veterans Affairs, Technology Acquisition Center Carolyn Carbone,** Contracting Officer

Email[: Carolyn.Carbone@va.gov](mailto:Carolyn.Carbone@va.gov)

Phone: 732-440-9742

**Susan Banasiak**, Contract Specialist Email: [Susan.Banasiak@va.gov](mailto:Susan.Banasiak@va.gov) Phone: 732-440-9694

# Submitted by: Nester Consulting, LLC d.b.a. GovernmentCIO

**Ross Aboff**, Vice President | VA Account Manager Email[: raboff@governmentcio.com](mailto:raboff@governmentcio.com)

Phone: 732-938-3225

T4NG Contract Number: VA118-16-D-1003

*This submission includes data that shall not be disclosed outside the Government and shall not be duplicated, used, or disclosed-in whole or in part-for any purpose other than to evaluate this proposal. If, however, a contract is awarded to this offeror as a result of-or in connection with-the submission of this data, the Government shall have the right to duplicate, use, or disclose the data to the extent provided in the resulting contract. This restriction does not limit the Government's right to use information contained in this data if it is obtained from another source without restriction*

**TABLE OF CONTENTS**

[Technical Volume 1](#_TOC_250006)

1. [Team GCIO’s Technical Approach to Developing a Service Layer to Emulate Computerized Patient Retrieval System (CPRS) Remote Procedure Calls (RPCs) for Select Data Read Functions (RTEP B.1.1; PWS 5.2.1) 3](#_TOC_250005)
2. Team GCIO’s Technical Approach to Developing a Service Layer to Emulate CPRS RPCs for Select Data Read/Write Transactional Functions for Outpatient Pharmacy Computerized Physician Order Entry (CPOE) (RTEP B.1.2; PWS 5.2.2) 6
3. [Team GCIO’s Technical Approach to Providing Automated Testing for All Emulation, Including Comprehensive Regression Test Suite (RTEP B.1.3; PWS 5.5.5)](#_TOC_250004)

................................................................................................................................................8

1. [Team GCIO’s Technical Approach to Providing Initial Operating Capability (IOC) Support to Demonstrate that CPRS is Able to Retain Full Functionality Against a Single Centralized Service, Replacing those Functions of the Original, De-Centralized VistA Source Instances (RTEP B.1.4; PWS 5.2, 5.6) 9](#_TOC_250003)
2. [Team GCIO’s Technical Approach to Progressing Our Solution to National Deployment (RTEP B.1.5; PWS 5.7) 11](#_TOC_250002)
3. [Estimated Level of Effort (RTEP B.1.6) 11](#_TOC_250001)

TABLE OF FIGURES

Figure 1: High-Level Overview of To-Be Implementation 1

Figure 2: Organization Chart 2

Figure 3: Agile Scrum Framework 2

Figure 4: GMV RPCs 3

Figure 5: PCE RPCs 4

Figure 6: Architecture Model 4

Figure 7: As-Is 5

Figure 8: To-Be 5

Figure 9: Microservice Architecture 6

Figure 10: Features and Benefits of Team GCIO's Approach 6

Figure 11: Team GCIO's Approach to RTEP Requirements 7

Figure 12: VistA Sites Gateway with Service Layer 10

[Figure 13: Team GCIO’s Level of Effort 12](#_TOC_250000)

# Technical Volume

GovernmentCIO (GCIO), an SDVOSB, VOSB, and HUBZone-certified company, respectfully submits our response to T4NG-0250: VistA Adaptive Maintenance (VAM). We provide thought leadership and IT service excellence to the Federal government and Department of Veterans Affairs (VA), specializing in Agile best practices, software development and IT project management. GCIO is currently developing and deploying the VistA Surgery GUI application supporting web services and the integration with VistA. GCIO is a primary architect of the Veteran-focused Integration Process (VIP) and has supported numerous VIP implementation projects for software development tasks.

GCIO has partnered with dbITpro, who designed, developed, tested, and deployed the Chemotherapy Ordering Management System (COMS) for the Veterans Health Administration (VHA) Innovation Team. Through the VistA Evolution API Exposure project, dbITpro developed and deployed 226 web services for OI&T, providing a mechanism for access through the Electronic Messaging Interface (eMI). dbITpro provides development support on the VistA Surgery GUI and application enhancements and is currently modifying the VA VANTS application to provide SSOi and 2FA capabilities.

Team GCIO has developed web services for retrieving data from VistA since 2011. Our experience of successful implementation began with the development of the Chemotherapy Ordering Management System where we developed web services to read and write data using RPCs that CPRS currently uses to perform functions such as orders, vitals, patient information, allergies, notes, and including Patient Data Entry and Pharmacy Computerized Physician Order Entry (CPOE) functions. These services were written in Node.js and are used to access and retrieve data from VistA as a service layer. In addition, Team GCIO performed the VistA Evolution API Exposure 1.0 project where we wrote 226 web services using Java that called the RPC Broker through VistALink to read and write data to VistA. Most recently, Team GCIO has developed the initial prototype of the VistA Surgery User Interface for Tracking and Efficiency (SUITE) web application using web services written in JavaScript and Node.js to execute the RPC Broker to read and write data to VistA. All of these implementations demonstrate our knowledge, understanding and integration experience necessary for the VistA Adaptive Maintenance to provide the transformation mechanism to help move CPRS to be independent of MUMPS and set the foundation for the VA transition to a commercial EHR.

A high-level overview of our solution involves the development of Node.js services that will use the InterSystems Caché API to read and write data elements to the Globals within VistA. As depicted in **Figure 1**, CPRS would use the Node.js service layer to access the Caché API.

# Figure 1: High-Level Overview of To-Be Implementation

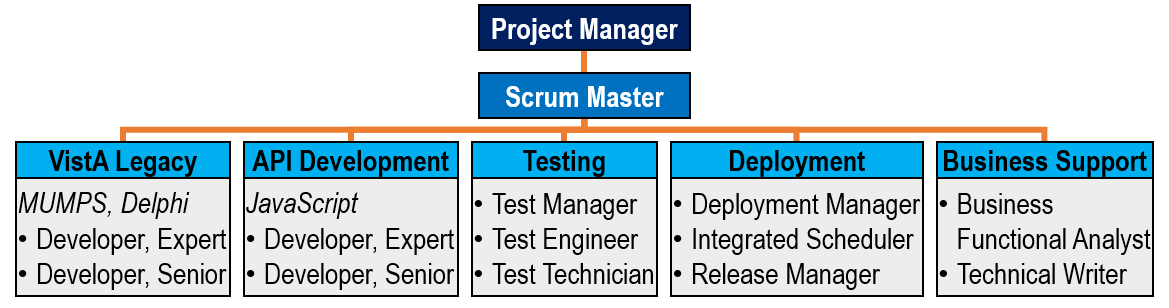


This approach will completely bypass the RPC Broker and Routines and will allow for CPRS to operate without the dependency of MUMPS code. These services will be written to mirror the current set of RPC data calls in use by CPRS. Team GCIO will develop these services calls using Fast Healthcare Interoperability Resources (FHIR) JSON format to permit these services to be shared with commercial EHRs.

Below is a high-level organization chart to show the Development Teams, Testing, Deployment, and Business support members that will help deliver VAM. The Development team will be split

into two groups, one Legacy team using MUMPS and Delphi code and a JavaScript web service development team to create all the services in Node.js and with Caché API. The Testing Team will perform all testing events against the developed code for the Node.js API and the integration and regression testing of services and CPRS. The Deployment team will manage the configuration and all releases of code throughout development to production environments. All of our team members are supported by the Business Support team composed of Business Functional Analyst and Technical Writers to assist with Rational and document management. **Figure 2** is a high-level organization chart for VAM.

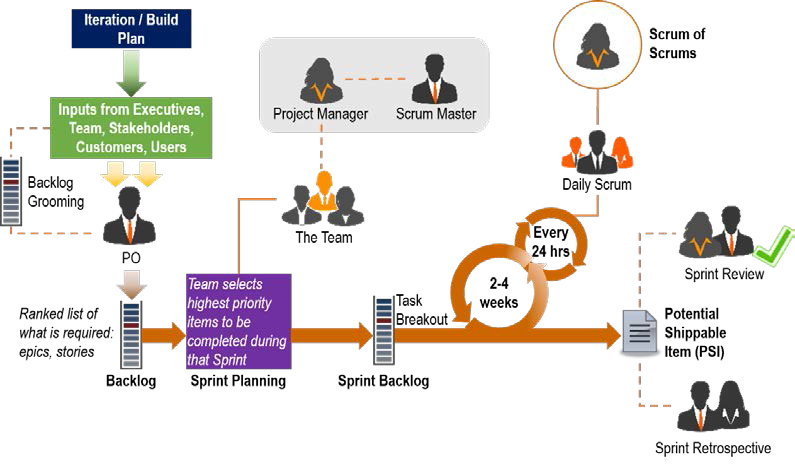
# Figure 2: Organization Chart



GCIO is assessed at CMMI-DEV maturity level 3 across our corporate Agile portfolio and certified in ISO 9001:2015. Our Development process is based on CMMI, ISO and ITIL quality and development best practices and is continually improved through our experience providing software development services for our numerous VA clients.

Team GCIO currently uses the Agile Scrum framework to deliver functionality for other VistA development efforts (e.g. VistA Surgery, Drug Accountability Upload (DAU)), using two-week sprint iterations and releasing new functionality and defect fixes into the production environment at the end of each sprint. **Figure 3** shows Team GCIO’s high-level Agile Scrum approach and incorporates the elements of the Scrum framework and the VIP methodology.

# Figure 3: Agile Scrum Framework



This approach demonstrates how customer-driven, prioritized requirements are broken down into short development sprints that result in reviewable, potentially shippable items. This approach allows VAM users to use and evaluate the system early and frequently throughout the overall schedule. High-priority sub-epics and user stories—whether due to risk assessment, business value, or other reason—are addressed early on to provide the result desired by the customer and give ample time for refinement of the product requirements. Agile development methodologies help attack risk through demonstrable progress; we use frequent, executable releases that enable and allow continuous customer involvement and feedback. Since all iterations end with a demonstrable release, the development team stays focused on producing results, and frequent reviews checks help ensure that the project stays on schedule. We apply this same approach to manage and deliver more than 40 concurrent software projects at the VA’s FSC.

# Team GCIO’s Technical Approach to Developing a Service Layer to Emulate Computerized Patient Retrieval System (CPRS) Remote Procedure Calls (RPCs) for Select Data Read Functions (RTEP B.1.1; PWS 5.2.1)

Our team is composed of a Senior Architect, Software Engineer, and MUMPS developers who will research each CPRS call to an RPC and the subsequent functions that occur after RPC execution in detail to identify input parameters, data variables, subsequent calls, and outputs. The team will analyze and design the current call being made to the RPC and determine which RPC and routines are being executed. The MUMPS developers will read the code and determine what routines are being executed and track down what data fields are being read or written to and will compose a data model of input and output data results. The team will identify the RPC, the INPUT PARAMETERS needed for execution and will create a RESTful service to access the VistA database Globals to read the requested data and formulate that response in JSON FHIR response to be returned to the CPRS client.

The team will create a suite of services that will be used to access the FileMan utilities and be able to navigate the core VistA features and functions of FileMan. The team will develop a system service to access the FileMan data dictionary and utilities for file manipulation, attributes, edits and generate a model using a Node.js service that will map the data dictionary and their correlation to Globals. The data model will be able to be defined by field, field type and value for each service. The team will compose and document the list of services developed to support the reading and writing of CPRS functions. These RPCs are from the file number 19, the OPTION file using the OPTION ‘OR CPRS GUI CHART’. The team will use a subset of these RPCs to develop the Node.js service layer. **Figure 4** is an example list of RPCs that will be developed as a service. The team will coordinate with the VA to ensure an accurate list is developed during our requirements elaboration period. **Figure 4** is a list of GMV calls made by CPRS.

# Figure 4: GMV RPCs

|  |  |
| --- | --- |
| **Patient Vitals Data Entry, Patient Allergy Data Entry, and Patient Problem Data Entry RPCs** | |
| GMV ADD VM | GMV GET VITAL TYPE IEN |
| GMV ALLERGY | GMV LATEST VM |
| GMV CLOSEST READING | GMV LOCATION SELECT |
| GMV CONVERT DATE | GMV MANAGER |
| GMV DLL VERSION | GMV MARK ERROR |
| GMV EXTRACT REC | GMV PARAMETER |
| GMV GET CATEGORY IEN | GMV USER |
| GMV GET CURRENT TIME | GMV V/M ALLDATA |
|  | GMV VITALS/CAT/QUAL |

**Figure 5** shows the list of calls made for Patient Care Encounter (PCE).

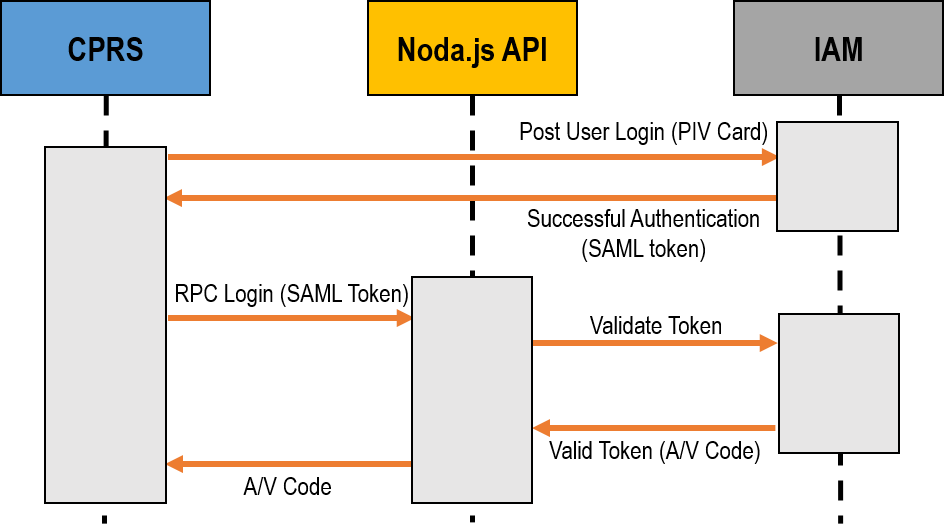
# Figure 5: PCE RPCs

|  |  |  |
| --- | --- | --- |
| **Patient Care Encounter RPCs** | | |
| ORWPCE ACTIVE CODE | ORWPCE GET EXCLUDED | ORWPCE LEXCODE |
| ORWPCE ACTIVE PROV | ORWPCE GET HEALTH FACTORS TY | ORWPCE LOADGAF |
| ORWPCE ACTPROB | ORWPCE GET IMMUNIZATION TYPE | ORWPCE MH TEST AUTHORIZED |
| ORWPCE ALWAYS CHECKOUT | ORWPCE GET SET OF CODES | ORWPCE MHCLINIC |
| ORWPCE ANYTIME | ORWPCE GET SKIN TEST TYPE | ORWPCE MHTESTOK |
| ORWPCE ASKPCE | ORWPCE GET TREATMENT TYPE | ORWPCE NOTEVSTR |
| ORWPCE AUTO VISIT TYPE SELECT | ORWPCE GET VISIT | ORWPCE PCE4NOTE |
| ORWPCE CPTMODS | ORWPCE GETMOD | ORWPCE PED |
| ORWPCE CPTREQD | ORWPCE GETSVC | ORWPCE PROC |
| ORWPCE CXNOSHOW | ORWPCE HASCPT | ORWPCE SAVE |
| ORWPCE DELETE | ORWPCE HASVISIT | ORWPCE SAVEGAF |
| ORWPCE DIAG | ORWPCE HF | ORWPCE SCDIS |
| ORWPCE FORCE | ORWPCE HNCOK | ORWPCE SCSEL |
| ORWPCE GAFOK | ORWPCE I10IMPDT | ORWPCE SK |
| ORWPCE GAFURL | ORWPCE ICDVER | ORWPCE TRT |
| ORWPCE GET DX TEXT | ORWPCE IMM | ORWPCE VISIT |
| ORWPCE GET EDUCATION TOPICS | ORWPCE ISCLINIC | ORWPCE XAM |
| ORWPCE GET EXAM TYPE | ORWPCE LEX | ORWPCE1 NONCOUNT |
|  |  | ORWPCE4 LEX |

Each service’s response will be structured into two distinct sections that will define the FHIR response for sharing with HL7 and external EHR systems and a VA-centric side of data elements. The VA centric data elements will be created and designed to specifically categorize VA-related data that would not relate to external EHR systems. This approach will create a JSON response to be a distinct yet a unified data model of VA and non-VA data that will be free from MUMPS dependency.

Our approach will be written in JavaScript and will be deployed to Node.js server. This architecture in **Figure 6** depicts the login functionality between CPRS, IAM, and Node.js. The model shows the use of a user’s PIV Card and A/V code with Node.js, JSON Web Token (JWT) and VistA. The services will also use Security Assertion Markup Language (SAML) for SSO authentication with the VA Active Directory.

# Figure 6: Architecture Model

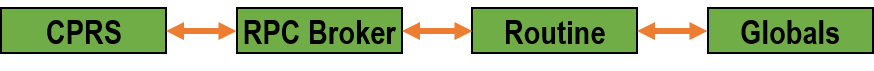


Our team will have Delphi and MUMPS developers to ensure a complete understanding of the back-end functionality and a thorough review of each RPC. In addition to our approach of using the RPC Broker help option to return which RPCs are called, the team will use developers to read and validate the code. A deep MUMPS analysis will benefit the development team with the identification of processes that occur post execution of the RPC with subsequent routines and will ensure total functionality is mimicked in the new service call. This analysis will prove fruitful when performing regression testing of the requirements to ensure no functionality is lost and that backward compatibility will remain intact. Team GCIO’s approach uses JSON modules for data structure and a NoSQL data storage approach.

Team GCIO has worked with both the Joint Legacy Viewer and the current Virtual Patient Record (VPR) and has used both in development environments to retrieve data from VistA. These technological outcomes can be incorporated in our service layer and ensure that security, auditing, and strict identification of data calls are logged and accurately captured. This method will ensure the capability to lock and restrict access to unauthorized service calls to VistA. Our experience with the VPR, COMS, and the integration of data using JavaScript and JSON responses with RESTful services clearly demonstrates our ability to successfully implement a Node.js MUMPS free service layer to read data functions for CPRS.

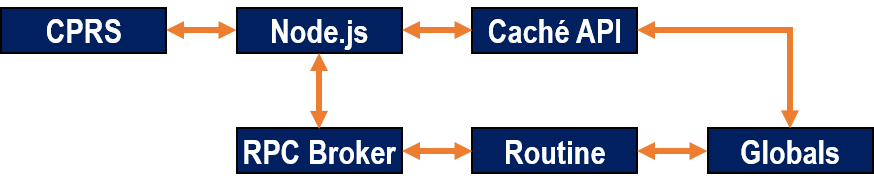
The current state is depicted in **Figure 7** shows a simple visual representation of the current data flow and integration points between CPRS and VistA.

# Figure 7: As-Is



As depicted in the diagram below, our solution will free the VistA data elements from MUMPS and at the same time maintain the current state of functionality. The Node.js server will act as a router to direct the service call to either the RPC broker or the Caché API as shown in **Figure 8**.

# Figure 8: To-Be



Our service layer will execute either the Caché API or the RPC Broker to access the data in the Global depending on service availability. This will allow for simultaneous use of the new services written for CPRS and the services that will be pending development. Our solution uses Microservices to respond to requests from the Node.js JavaScript API layer. The advantages of using a Microservice Architecture have been well documented in the private sector, with many well-known organizations moving towards a distributed service architecture. The VA would benefit from moving to a distributed architecture for these reasons:

* + **Fault isolation:** When a service fails, other services/modules are protected by isolation from failures affecting the whole system. No long-term commitment to a single technology allows

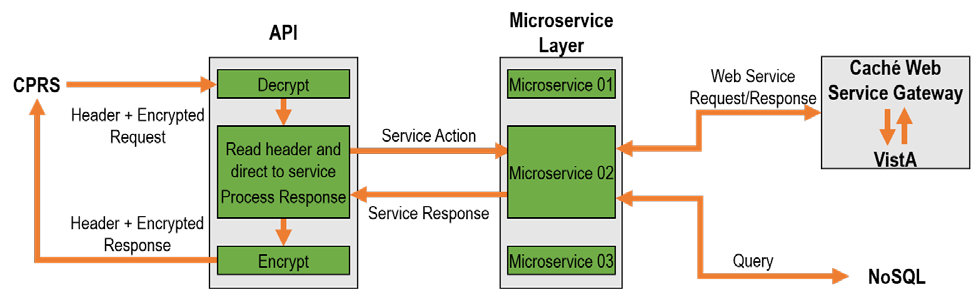
developers to use the appropriate tech stack to address a problem.

* + **Quicker on boarding:** New development teams can understand the purpose and structure of an isolated small code base faster than a monolithic application.
  + **Scalability:** The Veteran Integrated Care Services (VICS) layer would have the ability to be scaled by adding extra nodes to handle extra load, complexity or the addition of RPCs and calls that are made with CPRS.

Each service would handle a well-defined business function in isolated containers. Our solution would use a container management system, Docker, to run each service within its own isolated process. This will allow the VA to leverage virtualization to save costs, automate deployments, and leverage intelligent scaling logic to respond to varying system loads. An example of a well- defined business function within the VA is reading/writing Vitals for a patient. The Vitals Microservice will store data related to a patient in a NoSQL store along with a patient identifier. All requests to write or read Vitals will be processed through the Vitals Microservice. This approach allows the VA to scale its Vitals business function in isolation.

**Figure 9** depicts the flow of services between CPRS and the Node service layer.

# Figure 9: Microservice Architecture



**Figure 10** shows the features and benefits of Team GCIO’s VAM approach and the overall benefit this effort will have to the VA’s transition to a commercial EHR. Team GCIO understands that this effort is to create the much-needed service layer for CPRS to position the VA to be flexible and able to migrate to an alternate data source without losing ingrained clinical work.

# Figure 10: Features and Benefits of Team GCIO's Approach

|  |  |
| --- | --- |
| **Feature** | **Benefit** |
| ***Centralized Service Layer*** | The VA has attempted to create several service layers and all previous approaches focused on the service local to the instance. This approach will benefit VA because it focuses on the current services provided by CPRS, creates a universal layer for those services and position CPRS and VA to move destination data sources with ease. |
| ***RESTful services*** | The VA will have RESTful services to call repeatable needs that are used in CPRS and permit the data exchange between a client, server, and an application. |
| ***Fast Healthcare Interoperability Resources*** | The use of services and returning the result in FHIR JSON format will permit data exchange with entities within and external to the VA. This will be another step to ensure the services developed will permit easy integration with a commercial EHR. |
| ***CPRS Positioning*** | CPRS is the main clinical application used throughout the VA. This service layer will allow VA to continue to operate CPRS while a commercial EHR is configured and deployed. Upon completion, the VA will be able to continue to use CPRS thanks to VAM creation of this service layer with Node.js |

1. **Team GCIO’s Technical Approach to Developing a Service Layer to Emulate CPRS**

**RPCs for Select Data Read/Write Transactional Functions for Outpatient Pharmacy Computerized Physician Order Entry (CPOE) (RTEP B.1.2; PWS 5.2.2)**

As with our approach in **Section 1**, Team GCIO will use our experience and capability to develop and implement Node.js services that write data to the VistA data source. Our experience through COMS, SUITE, and VistA Evolution API Exposure 1.0 were all web services that performed Read/Write functionality. Our team will create a service layer using Node.js and Caché web services to access and update data elements of VistA without the use of MUMPS. The team will research and document the details needed to call RPCs in CPRS to order a variety of items and activities including medications, lab tests, and consultations. The team will then recreate those calls using the Caché Web Service and will perform the read\write capability needed. The team has already created web services for RPCs to use pharmacy, labs, and radiology departments and would be able to adapt that code to the creation of these services. We will then validate the data returned and will execute numerous test cases to perform unit, integration, and regression testing of the source code and CPRS.

The team will use JavaScript, Node.js, and JSON to script web services to create the VICS layer for the Outpatient Pharmacy functionality of CPRS. The team will create, separate, and categorize the services by CPRS modules and the VistA OPTION file settings to set familiarity to downstream developers who will access and use these services in the modernization of VistA. As in the previous section, the team will document the services, the code and will use Rational to conduct requirements analysis, development artifacts, testing, and validation. The team will modify the service to follow the architecture diagram described in **Figure 6**. The team will create services to access FileMan using JavaScript and Caché Web Services to create a data model for Computerized Physician Order Entry (CPOE). The team will work with Business Owners and Stakeholders to identify the services needed to be replicated and will work with and identify the data elements to be used for testing and acceptance.

Our team has worked extensively with VistA Pharmacy through our COMS and SUITE efforts and have strong working knowledge of Pharmacy and the need for validation of prescriptions and drugs for patients. The team will work with the RPCs under the Pharmacy package to create the service layer for Outpatient Pharmacy and to provide the flexible querying needed. The team will create a web interface that will use these services to capture, submit and display data to the pharmacy users in a modern view. The team has used services that check drugs for drug-to-drug interactions, adverse reactions, and other checks that are in place today to protect the patient from harmful mixtures. The team has extensive experience designing, developing, and integrating these services for COMS and SUITE applications that the team could convert these services from calling RPCs to call a Caché web service that would read/write data to the same source without MUMPS. The team will perform regression testing on the unit and integration tests for CPRS and VistA Roll and Scroll (VSRS) to ensure backward compatibility between existing RPCs and the new VICS layer.

**Figure 11** defines at a high-level Team GCIO’s solution for each RTEP requirement.

# Figure 11: Team GCIO's Approach to RTEP Requirements

|  |  |
| --- | --- |
| **Requirement** | **Solution** |
| ***FileMan data modeling using web- standard technologies and representation*** | Our team will create a Node.js service to call Caché APIs to access the FileMan files within the VistA database and will create a data model that can be used to query and perform FileMan functions to the VistA database. |

|  |  |
| --- | --- |
|  |  |
| **Requirement** | **Solution** |
| ***Distinguishing VA-specific from generic healthcare patterns*** | The team will develop a JSON response using three data sections for parsing and disseminating to external partners. The JSON response will use VA and Non-VA data and will use FHIR response to return EHR data in JSON and HL7 format. |
| ***Implementing MUMPS emulation using JavaScript/Node.js-driven, model-driven replacement.*** | Through the use of Caché APIs and a service layer using Node.js, the team will create a RESTful service based call using JavaScript on Node.js server. |
| ***Comprehensive analysis of logic of CPRS client and its RPC interfaces*** | The team will use a MUMPS and Delphi developer to research code execution and determine the subsequent actions taken after an RPC is called to track all functionality. |
| ***Operationalizing JSON models on NoSQL data stores*** | Our approach natively uses JSON and NoSQL data stores to send and receive data between stores. |
| ***Analysis of JLV and its VPR interface*** | Our Developers will review the code and services currently in place with the JLV and\or VPR to format and reuse JSON outputs from those services. |
| ***A final solution that has no legacy MUMPS dependencies.*** | Our solution is completely independent and will not use any MUMPS code to perform the function called and will read data from VistA. |

# Team GCIO’s Technical Approach to Providing Automated Testing for All Emulation, Including Comprehensive Regression Test Suite (RTEP B.1.3; PWS 5.5.5)

Team GCIO will use an Agile development test methodology whereas each requirement is tested during the development to ensure that the developed code can meet automated testing of CPRS and the Node.js service layer with our CI engine. The Testing Team is composed of a Test Manager, Test Engineer, and Testing Technician who will perform various tests with CPRS and the VICS layer and the environment to ensure that the requirements are met, the performance is not hindered and the product security is not compromised. At the start of each effort, the Test Manager in conjunction with Senior Architect and Scrum Master will develop a Test Strategy and will incorporate the VA’s implementation of Rational Quality Manager tool within 15 days after the Technical Kickoff Meeting. The Test Strategy will identify the method of testing; the tools that will be used for automated and manual testing; and the goal, actors, and the roles for each test period.

The Testing Team will create and maintain the Test Plan, Test Cases, Test Data, Test Coverage Traceability Matrix, and Test Results in Rational to comply with VIP requirements. The team will conduct unit, functional, accessibility, system, reliability, usability, interoperability, regression, security, performance, and Non-Functional testing of the requirements. Each test will be executed and documented with the date of execution, test time, test duration, test name, test details, tester, and the test result.

Our Test Manager will create and update the Test Plan data in the Rational Quality Manager following the templates and data requirements. The Testing Team will enter in the relative information for all executed tests with Rational and will ensure that there is a linkage between requirement, test data, and test result. The team will document and generate a Test Results in the Rational Quality Manager which will serve as the final piece of data that completes the Requirements Traceability Matrix (RTM). Our testing team will work the OI&T security group to conduct a WASA and Fortify scan of each product and package updates. These scans will identify if any security holes exist within the product code and the results will be used to provide VIP reviews with documentation that proves the product is safe and secure for release. The team will use these tools and other 508 compliance tools that will scan the system and ensure compliance with security and accessibility.

The Testing Team will conduct Performance Testing of the CPRS and the VICS layer to ensure that the introduced changes do not affect the performance of the data retrieval from VistA. The team will work with the System Engineering Design and Review (SEDR) group to ensure that the data transmissions initiated by the Node.js service layer on the network do not adversely affect the performance of the VA network. All testing, test scripts, test documentation and test results will be documented and published to the Rational Tools Concert to provide utmost transparency of the testing process of the product and permit historically review of test and scan results.

As defects are identified, the team will detect the root cause of the defect, will document the actions taken to generate the defect and will either identify and deploy a fix for the item or will make a recommendation to the VA PM to be added to the product backlog. Team GCIO will then discuss the identified defect with the PM and provide a recommendation for the prioritizing of its repair. All defects, test cases, test scripts, test results and test documentation will be uploaded daily to Rational Team Concert by our Test Manager.

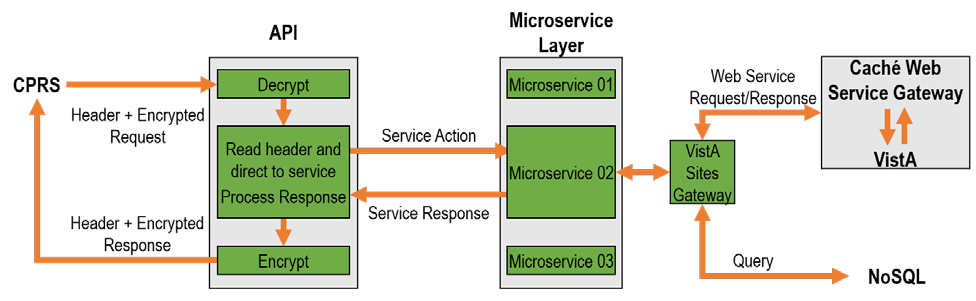
A feature of our testing process is the automated Continuous Integration testing of VICS with CPRS and generating the validation of those tests results with the RPCs. The team will use Rational and CI to import code changes, execute the build and push the code to a test environment. The CI engine will then execute the unit and integration test scripts that will be written for the existing CPRS requirement and the VICS layer requirement. This documentation will prove the data output and input results are synched and the VICS call truly mimics the existing CPRS call. The CI will execute regression test scripts against the CPRS and VICS layer to validate the existing service does perform all the subsequent data calls and complete the intended transaction in VistA without hindering existing functionality. The details captured in the creation of the service layer calls are critical to using a validation and acceptance of the regression testing executed against CPRS. This implementation is critical in maintaining operational use of CPRS as the VA transits to a commercial EHR solution.

# Team GCIO’s Technical Approach to Providing Initial Operating Capability (IOC) Support to Demonstrate that CPRS is Able to Retain Full Functionality Against a Single Centralized Service, Replacing those Functions of the Original, De- Centralized VistA Source Instances (RTEP B.1.4; PWS 5.2, 5.6)

The team will develop a routing interface similar to the existing CPRS option to select different VistA sites. This feature will allow the client to select the destination of the data requests and can specify the VistA address and port. We have implemented this approach in COMS and SUITE software using JavaScript and Node.js and could easily import this feature into this code base to create this service layer. This feature can be configured to be enabled or disabled when a central single instance of VistA is implemented.

The routing of service calls through the node.js service layer will be completed by building a parent services layer that will set the destination details from a pre-configured list. As depicted in our diagram below, the service layer would use a microservice that would be enabled to select a destination for the calls to be made. **Figure 12** shows the transmission process for a multi-site use.

# Figure 12: VistA Sites Gateway with Service Layer



The above approach demonstrates the ability for VICS to provide the dual functionality needed to maintain operational use of CPRS and the use of VICS layer. The model shows the data exchange between a single CPRS instance and several VA sites. The approach allows for the enabling and disabling of VistA Site Gateway service that will permit the destination selection of the data calls allowing sites to access different VistA instances. Once consolidation is complete, the feature will be disabled and all calls will be sent to the single instance.

Our team will support the IOC testing process by providing a Deployment, Release, and Configuration Manager to collaborate with our developers and business support staff to orchestrate the testing events needed with the IOC testing sites. Our team will identify the business stakeholders and will format and track the submission and acceptance of MOUs for IOC sites. In addition, our team will request the IOC site to designate a member to complete the **IOC Operating Entry Request and Exit Summary (PWS 5.6-D)**. Our team will generate the **Implementation Plan (PWS 5.6-A)** and **Operational Ready Review Checklist (PWS 5.6-L)** to identify the strategy, installation schedule, steps, configuration items and sites for IOC testing. The team will conduct meetings and demonstration with the IOC test site members and will facilitate the exchange of documents, scripts, and results between test site participants, OI&T, and business stakeholders. Our Support Team will create a **Defect Resolution Plan (PWS 5.6-B)** that will detail the data items to capture, policies, procedures and the defect resolution plans for review, implementation, acceptance, and release. Our team will document in through detail any Defects identified and will categorize them by site using **IOC Defect Tracking Spreadsheet (PWS 5.6-**

**C)** and an **IOC Defect Log (PWS 5.6-E)**. The project manager will submit to the VA COR and PM a **Defect Status Report (PWS 5.6-H)** five days after the start of the IOC event outlining the number of defects opened and closed and a brief summary of the impact and changes. The team will capture the user, error, steps taken and any information that may be useful to the developer in identifying the cause of the problem. The team will capture and generate **IOC Evaluation Summary (PWS 5.6-F)** that will identify the success of the overall testing event, the number of defects identified, resolutions implemented and an **IOC Execution Log (PWS 5.6-G)** for the collection of sites.

If a release is necessary to implement resolutions to defects, the team will create a **Package/Patch Completion Transition Document (PWS 5.6-I)** that will be used to track and disseminate information regarding changes to the VICS layer. The team will draft the **Lessons Learned Report (PWS 5.6-J)** to contain any items the team discovered during this effort and will categorize the business category that they relate to. Upon completion of the IOC testing and approval that the product works to specification, the team will submit a complete **National Release Checklist**

**(PWS 5.6-K)** and a **Deployment Plan (PWS 5.6-M)** to outline the steps necessary for National Deployment. The plan will identify the deployment rate, schedule, actions and will include any rolls back, installation steps and configuration changes to hosting or server software or setup. The team will provide the complete **Software Code Package (PWS 5.6-N)** to include the code changes for Node.js, a complete npm to support VICS, the code changes for the Caché web services, and any unforeseen changes that are needed to FileMan or Globals to support the successful implementation of VICS.

# Team GCIO’s Technical Approach to Progressing Our Solution to National Deployment (RTEP B.1.5; PWS 5.7)

Team GCIO’s Business Functional Analyst, Deployment, and Release Manager will draft and deliver a **Deployment, Installation, Back-Out, Rollback Plan (PWS 5.7-A)** for the implementation of this service layer. The team will use the VA’s format and will author a compliant document for release. The release notes will identify a brief overview of the product, the supported RPCs and program functionality that is implemented. The document will discuss in detail the deployment and installation steps necessary at a technical level that will provide clear and concise direction. The team will provide support and operational guidance to the Enterprise Service Line with the release of this software.

The team will work with VA to establish a release of the product to the VA’s cloud. The team will prepare and submit the documentation required, the team will coordinate the calls and approvals needed to release the product to the VA’s cloud. Our team will document the VICS layer in detail using the VIP documentation. The team will work with the POLARIS scheduling tool to enter in events, dates, times, resources, and other details needed to schedule the release of the software throughout the VA.

All of our documents and process will adhere to the VA’s VIP. Our team has two projects currently active in VIP and are very familiar with VIP and the implementation and impact the change has made on the VA. Our team is extremely familiar with and currently use Rational to support all of the requirements, compliance stories and epics needed for the product and VIP. Our team will work with the organizations identified in the VA that are necessary to achieve VIP success. All meetings will be documented and recorded in Rational.

The team will create a POM for the VICS layer and will identify administrative procedures, security, identity management, user notifications, system monitoring, reporting, tools, exception handlings, dependent systems, system recovery and troubleshooting procedures for VICS layer. The document will be provided in MS Word and Adobe Acrobat formats and will be posted to the VA’s Rational server.

The team will manage and conduct test site calls with each IOC production site. The calls will be held in an open forum and will permit the callers to raise issues, defects and other items related to the VICS deployment. The team will document all participants and will provide meeting minutes following the meeting that will capture in detail topics discussed and questions or issues raised. The team will keep a detailed log of these events and will report them to VA PM.

# Estimated Level of Effort (RTEP B.1.6)

In planning our approach to this task, we developed a detailed Work Breakdown Structure (WBS) to ensure appropriate allocation of resources to accomplish all PWS activities. The WBS decomposed the PWS requirements into precise work packages that define the activities and

associated resources and durations needed to complete every task and subtask and established the baseline for our level of effort (**Figure 13**).

# Figure 13: Team GCIO’s Level of Effort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PWS Task Area** | **Labo r Cate gorie s** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Project Manager** (GCIO) | **Integrated Schedule Analyst** (GCIO) | **Integrated Schedule Analyst** (dbITpro) | **508 Compliance Analyst** (GCIO) | **Technical Writer, Sr.** (dbITpro) | **Architect, Sr.** (dbITpro) | **Software Engineer, Sr.** (GCIO) | **Developer, Expert** (GCIO) | **Developer, Expert** (dbITpro) | **Senior Developer (Scrum Master)** (GCIO) | **Developer, Sr.** (GCIO) | **Developer, Sr.** (dbITpro) | **Business Functional Analyst, Sr.** (GCIO) | **Business Functional Analyst, Sr.** (dbITpro) | **Release Manager** (GCIO) | **Configuration Manager, Sr.** (dbITpro) | **System Administrator, Sr.** (GCIO) | **Database Administrator, Sr.** (dbITpro) | **Test Engineer, Sr.** (GCIO) | **Test Engineer, Sr.** (dbITpro) | **Test Manager** (dbITpro) | **Deployment Manager** (GCIO) | **Grand Total** |
| ***5.2*** |  | 960 |  | 160 | 960 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 2080 |
| ***5.2.1*** |  |  |  |  |  | 480 | 960 |  | 1920 |  | 1920 | 3840 |  |  |  |  |  |  |  |  |  |  | 9120 |
| ***5.2.2*** |  |  |  |  |  | 480 | 480 | 1920 |  |  | 1920 | 3840 |  |  |  |  |  |  |  |  |  |  | 8640 |
| ***5.2 Total*** |  | **960** |  | **160** | **960** | **960** | **1440** | **1920** | **1920** |  | **3840** | **7680** |  |  |  |  |  |  |  |  |  |  | **19840** |
| ***5.3 Total*** |  |  |  |  |  |  | **960** |  |  |  |  |  |  | **160** |  |  |  |  |  |  |  |  | **1120** |
| ***5.5*** |  |  |  |  |  | 480 | 480 |  |  |  |  |  |  |  | 480 | 480 |  |  |  |  |  |  | 1920 |
| ***5.5.1*** |  |  |  |  |  | 480 | 480 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 960 |
| ***5.5.2*** |  |  |  |  |  |  |  |  |  | 480 |  |  |  |  | 240 | 240 |  |  |  |  |  |  | 960 |
| ***5.5.3*** |  |  |  |  |  |  |  |  |  | 480 |  |  |  |  |  |  |  |  |  |  |  |  | 480 |
| ***5.5.4*** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 960 | 960 |  |  |  |  | 1920 |
| ***5.5.5*** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1920 |  | 1920 |  | 3840 |
| ***5.5.6*** |  |  |  |  |  | 480 | 480 |  |  |  |  |  | 480 |  |  |  |  |  |  |  |  |  | 1440 |
| ***5.5 Total*** |  |  |  |  |  | **1440** | **1440** |  |  | **960** |  |  | **480** |  | **720** | **720** | **960** | **960** | **1920** |  | **1920** |  | **11520** |
| ***5.6 Total*** |  |  |  |  |  |  |  |  |  |  |  | **960** |  | **480** |  |  |  |  |  | **480** | **480** |  | **2400** |
| ***Total (Base)\**** |  | **960** |  | **160** | **960** | **2400** | **3840** | **1920** | **1920** | **960** | **3840** | **8640** | **480** | **640** | **720** | **720** | **960** | **960** | **1920** | **480** | **2400** |  | **34880** |
| ***Total OY 1\**** |  | **960** |  | **160** | **1920** | **1440** | **2400** |  |  | **2400** |  |  |  | **2560** | **2400** | **2160** | **960** | **960** | **1920** | **480** | **2400** | **480** | **23600** |
| ***5.8***  ***(Optional Task)*** | 320 |  | 320 |  |  |  |  |  |  |  |  |  |  | 320 |  |  |  |  |  |  |  |  | 960 |
| ***Grand Total*** | **320** | **1920** | **320** | **320** | **2880** | **3840** | **6240** | **1920** | **1920** | **3360** | **3840** | **8640** | **480** | **3520** | **3120** | **2880** | **1920** | **1920** | **3840** | **960** | **4800** | **480** | **59440** |

\* - Per Discriminator B.1.6 in the RTEP, these hour totals are for PWS sections 5.2, 5.3, 5.5, 5.6, and 5.7 and all subparagraphs (5.X.X).