**TAC-17-42242 / RTEP #T4NG-0250**

**Task Execution Plan**

August 17, 2017

*Solution*

*Innovation*

*Teamwork*

*Motivation*

*Leadership*

*Business*

*Project Strategy*

Department of Veterans Affairs Office of Information & Technology

Enterprise Program Management Office

**VistA Adaptive Maintenance**

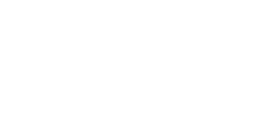
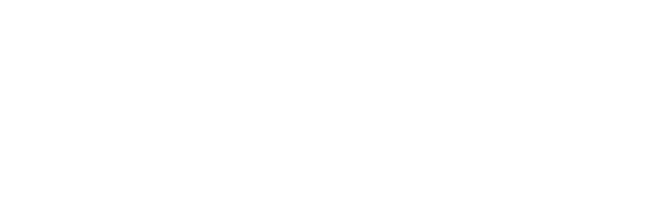
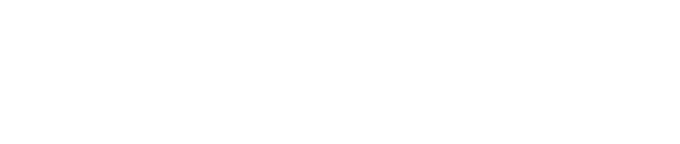
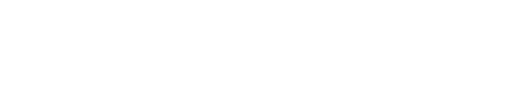
# Submitted to:

Carolyn Carbone, Contracting Officer

Department of Veterans Affairs, Technology Acquisition Center 23 Christopher Way

Eatontown, NJ 07724

# Submitted by:



Pro-Sphere Tek, Inc.

1101 King Street, Suite 200

Alexandria, VA 22314 Kevin D. Rice, President

Office: 703-810-3035, Mobile: 703-220-1453

[Kevin.Rice@prosphere.com](mailto:Kevin.Rice@prosphere.com)

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

*purposes only.*



# Table of Contents

[1.0 Introduction and Understanding 1](#_bookmark0)

* 1. [Technical Approach (RTEP B.1.1 and B.1.2) 1](#_bookmark2)
  2. [Developing a Service Layer to Emulate Computerized Patient Retrieval System (CPRS)](#_bookmark10) [Remote Procedure Calls (RPCs) for Select Data Read Functions (PWS 5.2.1, 5.2.1.1,](#_bookmark10) [5.2.1.2, 5.2.1.3) 6](#_bookmark10)
  3. [Developing a Service Layer to Emulate CPRS RPCs for Select Data Read/Write](#_bookmark14) [Transactional Functions for Outpatient Pharmacy Computerized Order Entry (CPOE)](#_bookmark14) [(PWS 5.2.2) 9](#_bookmark14)
  4. [Automated Testing of All Emulation, including Comprehensive Regression Test Suite](#_bookmark18) [(PWS 5.5.5, 5.5.5.1, 5.5.5.2) 10](#_bookmark18)
  5. [Initial Operating Capability (IOC) Support Demonstrating CPRS Retains Full](#_bookmark23) [Functionality against a Single Centralized Services Replacing those Functions of the](#_bookmark23) [Original, De-Centralized VistA Source Instances (PWS 5.2 and 5.6) 15](#_bookmark23)
  6. [National Deployment Following IOC so CPRS Retains Full Functionality against a](#_bookmark27) [Single Centralized Service Replacing the Functions of the Original 131 De-Centralized](#_bookmark27) [VistA Source Instances (PWS 5.7) 18](#_bookmark27)

[3.0 Level of Effort (RTEP B.1.6) 20](#_bookmark29)

List of Exhibits

[Exhibit 1: Team ProSphere 1](#_bookmark1)

[Exhibit 2: Team ProSphere’s Scrum Process 2](#_bookmark3)

[Exhibit 3: Agile Requirements Elaboration 3](#_bookmark4)

[Exhibit 4: Example VIP Compliance Epics Categories 3](#_bookmark5)

[Exhibit 5: Requirements Linkage 4](#_bookmark6)

[Exhibit 6: Proposed High Level Integrated Architecture 5](#_bookmark7)

[Exhibit 7: VistA Adaptive Maintenance Design Principles 5](#_bookmark8)

[Exhibit 8: Node.JS Common Components/Features 6](#_bookmark9)

[Exhibit 9: Node.JS Components 7](#_bookmark11)

[Exhibit 10: Delphi Components 8](#_bookmark12)

[Exhibit 11: MUMPS Component Modifications 8](#_bookmark13)

[Exhibit 12: Node.JS Components 9](#_bookmark15)

[Exhibit 13: Delphi Components 10](#_bookmark16)

[Exhibit 14: MUMPS Component Modifications 10](#_bookmark17)

[Exhibit 15: Incremental Test Approach 11](#_bookmark19)

[Exhibit 16: Testing Event 11](#_bookmark20)

[Exhibit 17: Patch Checklist 14](#_bookmark21)

[Exhibit 18: Team ProSphere’s Automated Testing Approach 14](#_bookmark22)

[Exhibit 19: IOC Checkpoint Requirements 15](#_bookmark24)

[Exhibit 20: How IOC Fits into the VIP Process 16](#_bookmark25)

[Exhibit 21: IOC Phases and Implementation 16](#_bookmark26)

[Exhibit 22: Deployment and Retirement Tasks 18](#_bookmark28)

[Exhibit 23: Level of Effort 20](#_bookmark30)

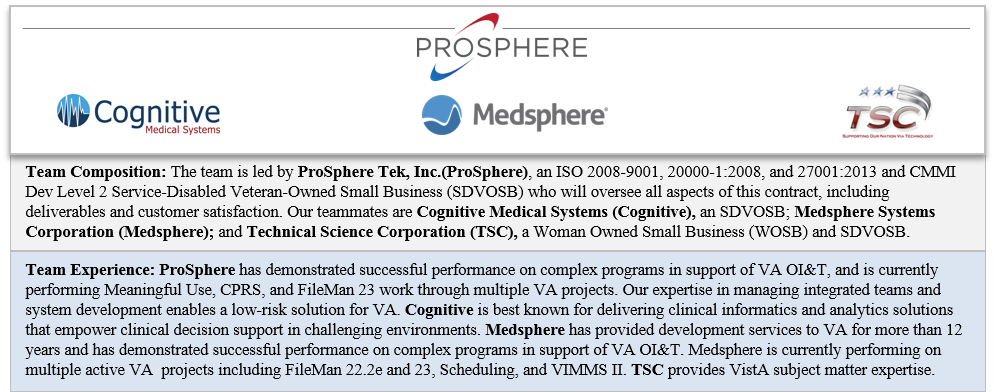


**1.0 Introduction and Understanding**

Team ProSphere provides an approach that combines our Agile Software Development Life Cycle (SDLC) methodology with Department of Veterans Affairs (VA) VIP processes in the execution of all tasks in the development of VistA Adaptive Maintenance (VAM). We understand the challenges, problems, and difficulties VA faces by having 131 separate, non- standard (VistA) instances and databases across its enterprise. We also understand the pathway forward for VA is its vision of an enterprise-wide view of patient data, interoperable Electronic Health Records (EHR), and a Service Oriented Architecture (SOA)-based set of common services for internal and external use. The clinical component of VistA is the Computerized Patient Care System (CPRS). This project isolates specific clinical components of VistA/CPRS from the non-clinical VistA and migrates these components to a secure, centralize service. Team ProSphere’s highly feasible approach provides the VA a proven low risk approach to providing a secure service based foundation to more easily migrate to a solution enterprise wide solution without harming or legacy systems and applications.

Team ProSphere consists of ProSphere, Cognitive Medical Systems, Inc., Medsphere Systems Corporation (MSC), and Technology Science Corporation (TSC). **Exhibit 1** identifies Team ProSphere’s composition and assignment of activities. Our core technical staff has many years of experience and in-depth knowledge of all aspects of this program including VistA, CPRS, FileMan and Massachusetts General Hospital Utility Multi-programming system (MUMPS).

# Exhibit 1: Team ProSphere



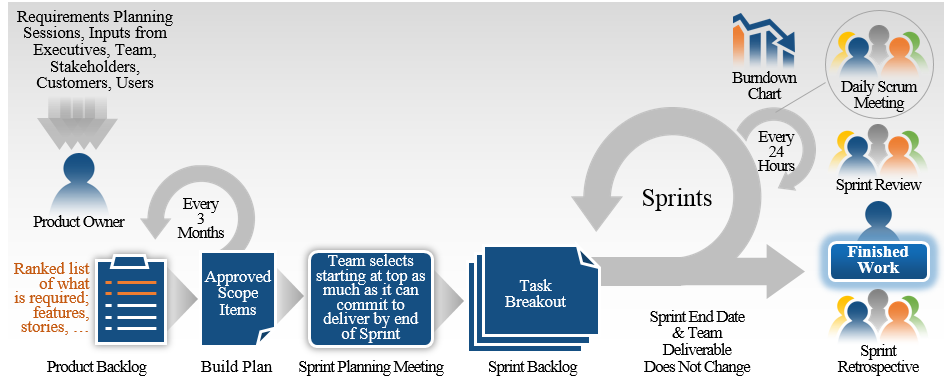
**2.0 Technical Approach (RTEP B.1.1 and B.1.2)**

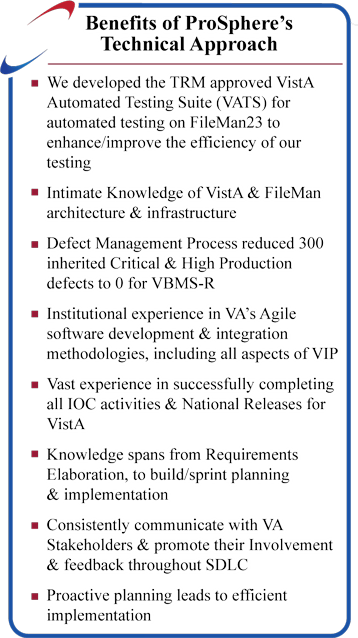
Team ProSphere’s technical approach for the successful execution of this Task Order (TO) builds on lessons learned from the successful implementation of the approach for other VistA and CPRS based programs such as FileMan23. We use Test Driven Development (TDD) in conjunction with a VIP-Centric Scrum implementation of Agile methodology (**Exhibit 2)** that uses six two-week sprints and produces new software releases every three months to provide adaptive maintenance and sustainment for PDE and CPOE. We use the first 2 sprints to perform the Planning Phase while using the remaining 4 sprints to implement the solution during the Build Phase. Beyond the strength of our processes, we are well-versed with industry-standard



Agile tools, including the IBM Rational Tool Suite. These tools support a lean and effective Agile development, and provide stakeholder visibility throughout the development life cycle.

# Exhibit 2: Team ProSphere’s Scrum Process



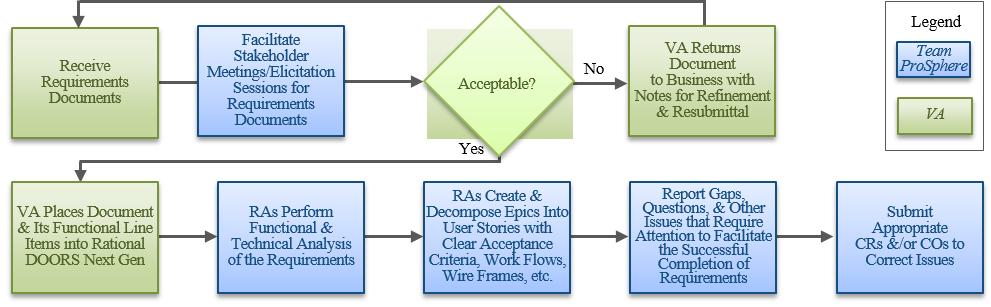
Our Scrum Master working with our PM remains in close and ongoing collaboration with the VA PM regarding the status of the planning and development for each release. They notify the VA PM of any risks and/or issues immediately, as well as providing them with a mitigation plan for the identified issues. At the conclusion of every sprint, our Scrum Master facilitates interactive Sprint Reviews with VA Stakeholders that showcases product demonstrations of the new capabilities delivered during the current sprint, and provides VA Stakeholders an opportunity to provide feedback to the development team.

Our development team uses their intimate knowledge of the PDE, CPOE, VistA, CPRS, and Fileman23 architecture to provide a Node.JS based service emulation layer to provide industry-standard, model- driven, secure service interfaces for new clients.

A quality product begins with a successful release that includes solid requirements and their corresponding LOE estimates to create an achievable scope. We used our experience providing full Software Development Life Cycle (SDLC) support for VBMS-R, FileMan23,

and CPOE to develop an Agile Requirements Elaboration process as shown in **Exhibit 3.** Following this process for every release helps prevent duplicated efforts by the Development Team ensuring on-time delivery of a robust, fully functioning application. We maintain all requirements in Rational DOORS Next Generation (RDNG) and ensure all requirement data are under change control and are fully linked to work items that show traceability to design changes, configuration items, test cases, and test results for the Release Manager (RM) assigned to the project.

# Exhibit 3: Agile Requirements Elaboration



Our Requirements Analysts (RAs) work with the VA PM and stakeholders to capture detailed business requirements for Epics, User Stories, Use Cases, and other sets of requirements. We continue our collaboration with VA stakeholders and technology professionals to begin decomposition of the business requirements into technical requirements that are captured in Epics and User Stories. Our RAs ensure that every Epic and User Story has clearly defined acceptance criteria, expected functionality, and any non-functional requirements to the level sufficient to support the design, development and testing phases. Included in the User Stories are any Wireframes, process flow diagrams, and/or interface descriptions required for successful development of the capability. Prior to the start of development for each release, our Scrum Master facilitates backlog grooming sessions with VA stakeholders to provide a review of, and agreement to, the Epics and User Stories to be included in the Build Plan. Team ProSphere’s RM ensures all Mandatory Compliance Epics (examples shown in **Exhibit 4**) and User Stories are properly entered in RDNG and associated to the current Build Plan for execution. Our experience has taught us that each project is different and that the VIP Mandatory Compliance Epic Categories are determined by the Program’s “3 In The Box” and VA Portfolio Manager.

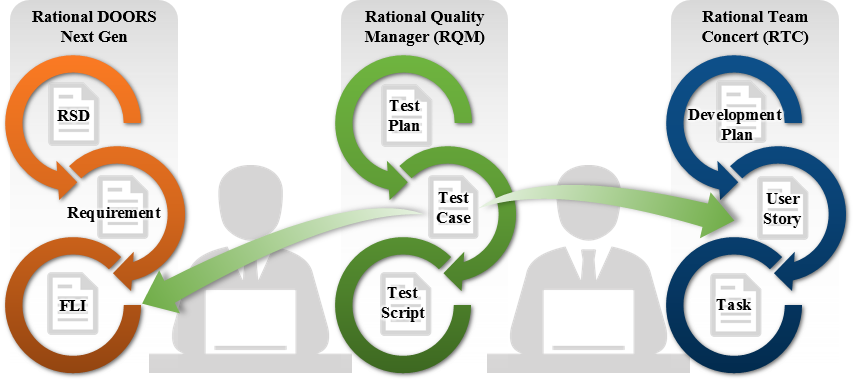
# Exhibit 4: Example VIP Compliance Epics Categories

|  |  |
| --- | --- |
| **Compliance Epic Categories** | **Description** |
| 508 Compliance | Ensures that the application being delivered to production has passed its Dequeue scans without any major or critical issues, and that it has received its Compliance Certificate and endorsement from VA’s 508 office. |
| Design, Engineering, and Architecture | Ensures all Service Level Agreements (SLA) and Operational Level Agreements (OLA) are approved and documented. |
| Testing and Defects | Ensures test plans and cases are specified and traced to release applicable user stories and work efforts via the Master Test Plan (MTP) and Requirements Traceability Matrix (RTM). Ensures that all current and past test cases have been executed and the results captured via the Test Execution Log (TEL). |
| Authority to Operate (ATO) | Ensures that the application has secured an ATO for the production environment. |
| Agile Process | Ensures all release applicable, approved elaborated business epics, sub-epics, and user stories of the solution are clearly scoped, defined, verifiable, and supportable by OI&T and the product owner. Ensures there are zero (0) high exposure open product risks that are not being monitored via the Risk Log. |
| Release Documentation | Ensures product deployment, installation, back-out, and data rollback procedures are specified via the Installation Guide. Ensures Version Description Document (VDD), User Guide, and Production Operations Manual (POM) are present, approved, and correct. |

We use our extensive knowledge, training, and experience with the Rational Tool Suite to ensure all requirements, Epics, User Stories, visualizations, test cases, stakeholder needs, requirements, and any other required artifacts are stored and linked properly. We use Rational Team Concert (RTC) to input and maintain all Epics and User Stories. We use Rational Quality Manager (RQM) to input and maintain all test cases and their corresponding results. These test results are coalesced in the Test Evaluation Log (TEL), a required artifact for the RA’s release readiness review. All requirement data is managed and stored in RDNG. We use the linking capabilities

within Rational to ensure that each requirement is linked, see **Exhibit 5,** to the appropriate Epic, User Story, work item, applicable test case(s) and results used to provide the required capability.

# Exhibit 5: Requirements Linkage



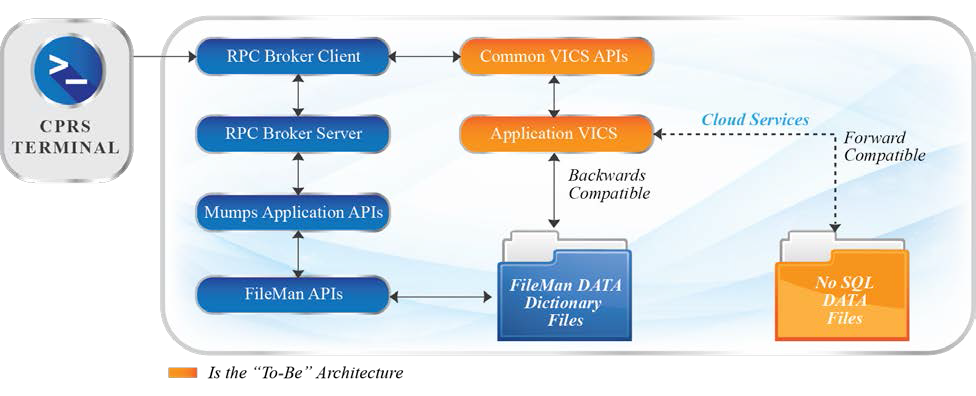
We use the “Upstream and Downstream Traceability” feature within RDNG to view the RTM graphically or in table format. The RTM traces all internal functional, non-functional and technical tests back to their given requirements. Through this traceability view, we determine how information is related, and identify requirements that might have been dropped or requirements that have crept in during development with no links to higher level requirements (scope creep). For example, one indication that a requirement is unsatisfied is that it does not have links. We use filters within our RTM view to easily identify these requirements. We create, update, and assist in the maintenance and updating of the RTM, mapping the use cases, test cases, and test results once test execution is underway to ensure that all requirements are being met and stakeholders are informed. We use RDNG to maintain the RTM ensuring all requirements are properly and fully linked to work items, test case, and test results. We provide an export of the RTM for further review when required. We continue our iterative requirements elaboration and refinement, and backlog grooming, to continuously support the build and development methodology to ensure compliance with the Epics and User Stories identified from the backlog while ensuring work is complete in alignment with VA priorities to meet VA needs.

The first step in the migration is to design a service emulation layer to replace the existing CPRS RPC interfaces and to provide compatibility with web-friendly and standard interfaces through Representational State Transfer (REST) web services. These web services support all RPCs used by the PDE and CPOE Veteran Integrated Care Services (VICS) that are installed using Node Package Manager (NPM). We use the knowledge and experience gained from performing this same type of work on FileMan 23 Enterprise project to create an n-tier architecture design that considers and addresses potential issues that arise with centralized services including location, time management, and synchronization. Our design ensures the legacy application functions continue to perform as before, but against single (individual) instances of centralized services, permitting the retirement of the equivalent function in the 131 VistA systems.

Team ProSphere designs and implements an integrated micro-service architecture, **Exhibit 6**, to compartmentalize the various common web service functionalities (Logging, Session Management, Two-Factor Authentication (2FA), and Data Marshalling), each of the PDE (Vitals, Allergy and Problem), and the CPOE applications. This approach facilitates an "adaptive maintenance solution" for each of the business applications and future VistA applications which

may use the general web service functionalities. This is done without "reinventing the wheel," and with the benefit of increasing the scalability, interoperability, portability of these systems.

# Exhibit 6: Proposed High Level Integrated Architecture



We use the design principles shown in **Exhibit 7** as the backbone of our design.

# Exhibit 7: VistA Adaptive Maintenance Design Principles

|  |  |
| --- | --- |
| **Design Principle** | **How Team ProSphere Implements Principle** |
| Comprehensive Analysis of Logic of CPRS Client and its RPC Interfaces | Before our Development Lead (DL) begins the design process, our Requirements Analysts (RAs) collaborate with VA stakeholders, business owners, and other SMEs and utilize our Requirements Elaboration Process (Exhibit 3) to identify all affected business processes functioning in RPCs for Patient Vitals, Allergy and Problem Data Entry, and Outpatient Pharmacy CPOE applications. Our DL uses these requirements as the basis for designing and providing an adaptive maintenance solution for these VistA functions on an individual, unique basis. To ensure no use cases or functionality were missed during the requirements elaboration phase, our development team will collaborate with VA SMEs and users to perform a full analysis of the underlying RPC interfaces and MUMPS code for each of the existing logical paths. As each of these logical paths are analyzed, we create automated JavaScript (JS) Unit Test scripts which will validate each of the key logical transition points found within the existing logical paths. Our team uses our testing methodology defined in Section 2.3 to ensure proper implementation and testing of JavaScript code. We establish test environments, cases and scripts for testing the existing logic and functions as part of the analysis so that effective comparatives in behavior may be Baselined. |
| Implementing MUMPS Emulation using JavaScript/Node.js- driven, Model-driven Replacement | Once we complete our full logic path analysis, our Development Team uses our technical approach to create new Express.JS resources and Node.JS logic to replace the current Delphi and MUMPS logic and ensure that each of the logical transition points handle identical input/output values and behavior to ensure backward compatibility and prepare for forward compatibility.  The current CPRS implementation utilizes data provided by RPCs in a MUMPS format. To ensure backwards compatibility, our design incorporates these data structures, while providing forward compatibility by utilizing JavaScript Object Notation (JSON) data structures, allowing them to be easily consumed by industry-standard services and applications. We provide code consistency, simplicity of future enhancement and improve sustainment capabilities using Node.JS compatible enhanced object relational models (ORM), such as Mongoose. The ORM also provides abstraction functionalities (data abstract layer), for both backward and forward compatibility logical paths. |
| Distinguishing VA- specific from Generic Healthcare Patterns | Our design process considers and distinguishes differences between VA-specific and generic healthcare patterns when developing these models, including terminology standards such as SNOMED Clinical Terms (CT), Current Procedural Terminology (CPT), International Statistical Classification of Diseases and Related Health Problems (ICD) 9/10, Logical Observation Identifiers Names and Codes (LOINC), RxNorm, National Drug File (NDF) and other standards used and/or affected by the enhancements in the scope of this project. |
| Operationalizing JSON Models on NoSQL Data Stores | To operationalize these new JSON ORMs, our approach is to create a data access layer which can connect to an industry-standard database, such as MongoDB, while also providing backward compatibility by connecting to the existing local VistA instance. |
| A Final Solution that has No Legacy MUMPS  Dependencies | Our Development Team ensures that connectivity to the existing local VistA instances bypass any existing MUMPS business logic related to the PDE and CPOE applications, while ensuring safe and reliable data storage by using FileMan Data Dictionary files. By providing connectivity paths to both FileMan and industry-standard data stores, our highly feasible design provides a greatly increased ability to transition from legacy data stores to new data stores, greater scalability and extensibility, and facilitates potential data replication and/or migration in the future. Following our Testing Methodology (Exhibit 17) our Development Team oversees and takes the lead in the configuration of a VistA Test System (production clone) to validate interfaces to the new VICS and the existing CPRS. |
|  |  |

**VICS Common Micro-Services.** Our development team uses Microsoft® Visual Studio to create micro-services that provide common general functionalities for the VICS micro-services that contain the service layer emulations. Our team uses Karma/Jasmine or other One-VA Technical Reference Model (TRM)-approved framework to Unit Test all micro-services. Through our support for FileMan23 and eHMP, Team ProSphere has gained extensive insight and experience with the development, implementation, and release of Node.JS REST services that provides RESTful Application Programming Interfaces (APIs). Our development team utilizes this knowledge and experience to develop the required Node.JS component modifications **(see Exhibit 8)** and enhancements listed below.

# Exhibit 8: Node.JS Common Components/Features

|  |  |
| --- | --- |
| **Requirement** | **Components/Enhancements** |
| General Requirements for all Node.JS code | * Develop code so that it is extensible and configurable. * Develop code to provide JavaScript Object Notation (JSON) elements. * Develop code to provide Secure Socket Layer (SSL) communications. * Develop code to provide Performance Monitoring. |
| Create Common micro-services that provide general web service functionalities for the all distinct VICS micro-services | * Develop a micro-service that provides REST interfaces needed for logging   + Provide an interface that supports logging of application activities, user activities, errors and exceptions, and performance monitoring results   + Provide an interface that returns log information for the type of logging requested. * Develop a micro-service that provides REST interfaces to handle Session Management for micro-services that require user session management.   + Provide an interface that creates a new session and returns the new user session with associated application(s) and system(s).   + Provide an interface that returns an existing session for the request user, period, application and/or system.   + Provide an interface that updates an existing session and returns the updated user session with associated application(s) and system(s).   + Provide an interface that deletes an existing session when the user logs out and returns a log out response for each associated application and system. * Develop a micro-service that provides REST interfaces to handle Identity and Access Management (IAM) Single- Sign-On (SSO) functions.   + Provide an interface that authenticates a user session using the provided Security Assertion Markup Language (SAML) token and returns a new SSO session.   + Provide an interface that updates an existing SSO session and returns the updated SSO session.   + Provide an interface that deletes an existing SSO session when the user logs out and returns a log out response. * Develop a micro-service that provides REST interfaces to perform Data Marshalling.   + Provide an interface to transform M to JSON and return the appropriate JSON response.   + Provide an interface to transform JSON to M and return the appropriate M caret-separated string response. |
| General Requirements for all VICS micro-services | * Develop all VICS micro-services to use the Common micro-services for logging, session management, authentication and data marshalling). * Develop a data access layer to connect to the local VistA instance using InterSystems® Caché node adapter. * Develop a data access layer to connect to an industry-standard MongoDB (NoSQL) database using Mongo node adapter. |

Team ProSphere’s highly feasible approach provides the VA a proven low risk approach to providing a secure service based foundation to more easily migrate to a commercial cloud-based EHR solution enterprise wide without harming legacy systems and applications.

**2.1 Developing a Service Layer to Emulate Computerized Patient Retrieval System (CPRS) Remote Procedure Calls (RPCs) for Select Data Read Functions (PWS 5.2.1, 5.2.1.1, 5.2.1.2, 5.2.1.3)**

**Understanding.** Team ProSphere understands that the VistA Adaptive Maintenance project needs to migrate the Patient Vitals Data Entry, Patient Allergy Data Entry and Patient Problem Data Entry from VistA into distinct VICS, allowing for retirement of the equivalent existing functions. These new VICS (service emulation layer) will be developed as Node.JS micro- services that provide web-friendly interfaces for new clients while maintaining backward compatibility for CPRS, and addressing issues that arise with a centralized service including

location, time management, and synchronization.

**Feasible Approach.** Our development team uses our highly feasible approach previously described, and further detailed below, to migrate the PDE functions. Our development team creates an initial design of the new REST service emulation layer designed to replace the existing CPRS RPC interfaces based on the requirements defined in our elaboration process. These web services consist of an n-tier architecture design that considers and addresses potential issues that arise with centralized services including location, time management, and synchronization, while ensuring continued functionality of CPRS and allowing for the retirement of the equivalent functions in the existing 131 VistA systems.

**Analysis of the VPR RPC Interface.** The Patient Data Entries functions currently interact with the Virtual Patient Record (VPR) RPCs to ensure that the Joint Legacy Viewer (JLV) may present this information. Our RAs utilize our Requirements Elaboration process (**Exhibit 3**) to identify all affected business processes functioning in RPCs, including VPR, for a Vitals, Allergy, and Problem VICS to provide a service emulation layer which provides compatibility with web-friendly interfaces for new clients.

**Patient Data Entry (PDE) VICS Micro-Services.** Our development team uses Microsoft® Visual Studio to create all Node.JS REST components, and will Unit Test these components using Karma/Jasmine, or other TRM-approved framework. We leverage Team ProSphere's experience with development of Node.JS REST services that provides RESTful APIs for VA FileMan23 to develop the required Node.JS component modifications **(see Exhibit 8)** and enhancements listed below, **Exhibit 9**.

# Exhibit 9: Node.JS Components

|  |  |
| --- | --- |
| **Requirement** | **Components/Enhancements** |
| Create a distinct VICS micro- service to replace the Patient Vitals Data Entry logical paths currently provided by RPCs and MUMPS logic | * Develop a data abstract layer containing ORMs representing each data value output from the current RPCs which provides common syntax for function calls [(post, get, put, delete) or (create, read, update, delete)]. * Develop a data persistence layer that will cache non-personally identifiable data and is accessible to Node.JS as needed to improve performance and reduce network load. * Develop a business logic layer containing libraries for each existing business logical path which matches the same output as the corresponding RPC. * Develop an interface that will return a MUMPS formatted String response for each existing business logical path containing identical data elements to those currently returned by the corresponding RPC. * Develop an interface that will return a JSON formatted response for each existing business logical path containing identical data elements to those currently returned by the corresponding RPC. * Functional Test the corresponding REST interface to validate forward compatibility. * Regression Test the corresponding function in CPRS to validate backward compatibility. * Compare outputs from Baseline (tests of current CPRS functions), Functional (testing of REST interfaces) and Regression (tests of CPRS after rerouting) tests for each corresponding function to ensure that comparable functionalities have been produced. |
| Create a distinct VICS micro- service to replace the Patient Allergy Data Entry logical paths currently provided by RPCs and MUMPS logic | * Develop a data abstract layer containing ORMs representing each data value output from the current RPCs and provides common syntax for function calls [(post, get, put, delete) or (create, read, update, delete)]. * Develop a data persistence layer that will cache non-personally identifiable data and is accessible to Node.JS as needed to improve performance and reduce network load. * Develop a business logic layer containing libraries for each existing business logical path which matches the same output as the corresponding RPC. * Develop an interface that returns a MUMPS formatted String response for each existing business logical path containing identical data elements to those currently returned by the corresponding RPC. * Develop an interface that returns a JSON formatted response for each existing business logical path containing identical data elements to those currently returned by the corresponding RPC. * Functional Test the corresponding REST interface to validate forward compatibility. * Regression Test the corresponding function in CPRS to validate backward compatibility. * Compare outputs from Baseline (tests of current CPRS functions), Functional (testing of REST interfaces) and Regression (tests of CPRS after rerouting) tests for each corresponding function to ensure that comparable functionalities have been produced. |
| Create a distinct | * Develop a data abstract layer containing ORMs representing each data value output from the current RPCs which |
|  |  |

|  |  |
| --- | --- |
|  |  |
| **Requirement** | **Components/Enhancements** |
| VICS micro- service to replace the Patient Problem Data Entry logical paths currently provided by RPCs and MUMPS logic | provides common syntax for function calls [(post, get, put, delete) or (create, read, update, delete)].   * Develop a data persistence layer that will cache non-personally identifiable data and is accessible to Node.JS as needed to improve performance and reduce network load. * Develop a business logic layer containing libraries for each existing business logical path which matches the same output as the corresponding RPC. * Develop an interface that returns a MUMPS formatted String response for each existing business logical path containing identical data elements to those currently returned by the corresponding RPC. * Develop an interface that returns a JSON formatted response for each existing business logical path containing identical data elements to those currently returned by the corresponding RPC. * Functional Test the corresponding REST interface to validate forward compatibility. * Regression Test the corresponding function in CPRS to validate backward compatibility. * Compare outputs from Baseline (tests of current CPRS functions), Functional (testing of REST interfaces) and Regression (tests of CPRS after rerouting) Test each corresponding function to ensure that comparable functionalities have been produced. |

Our development team uses Embarcadero® Rapid Application Development (RAD) Studio™ to modify existing RPC Broker interfaces that utilize the newly developed REST interfaces instead of the MUMPS database. Our team Unit Tests these interfaces using DUnit. Our proposed Delphi Component modifications are listed in **Exhibit 10.**

# Exhibit 10: Delphi Components

|  |  |
| --- | --- |
| **Requirement** | **Component Modifications** |
| Create a distinct VICS micro-service to replace the Patient Vitals Data Entry logical paths currently provided by RPCs and MUMPS logic | * Modify each RPC Broker interface to perform an XMLHTTP call in order to consume the newly developed corresponding Vitals VICS REST interface.   + Ensure that changes are the minimal amount necessary to ensure proper connectivity to REST interface that passes same parameters currently passed to RPC and receives an identical output from the REST interface to that of the existing RPC.   + Ensure that authentication is updated to current SSO standards and passes sufficient user identification- related data to properly authenticate the user. |
| Create a distinct VICS micro-service to replace the Patient Allergy Data Entry logical paths currently provided by RPCs and MUMPS logic | * Modify each RPC Broker interface to perform an XMLHTTP call in order to consume the newly developed corresponding Allergy VICS REST interface.   + Ensure that changes are the minimal amount necessary to ensure proper connectivity to REST interface that passes same parameters currently passed to RPC and receives an identical output from the REST interface to that of the existing RPC.   + Ensure that authentication is updated to current SSO standards and passes sufficient user identification- related data to properly authenticate the user. |
| Create a distinct VICS micro-service to replace the Patient Problem Data Entry logical paths currently provided by RPCs and MUMPS logic | * Modify each RPC Broker interface to perform an XMLHTTP call in order to consume the newly developed corresponding Problem VICS REST interface.   + Ensure that changes are the minimal amount necessary to ensure proper connectivity to REST interface that passes same parameters currently passed to RPC and receives an identical output from the REST interface to that of the existing RPC.   + Ensure that authentication is updated to current SSO standards and passes sufficient user identification- related data to properly authenticate the user. |

Our Development Team uses InterSystems® Caché Studio to support the use of FileMan Data Dictionary files needed to ensure backward compatibility, and performs Unit Testing using VATS, the TRM-approved tool that Team ProSphere developed. **Exhibit 11** shows enhancements.

# Exhibit 11: MUMPS Component Modifications

|  |  |
| --- | --- |
| **Sample Task Requirement** | **Component Modifications** |
| Create a distinct VICS micro-service to replace the Patient Vitals Data Entry logical paths currently provided by RPCs and MUMPS logic | * Conduct analysis of any MUMPS logic currently used by the Vitals application * Add Entity file entries as required to support backward compatibility to FileMan data dictionaries and SSO security |
| Create a distinct VICS micro-service to replace the Patient Allergy Data Entry logical paths currently provided by RPCs and MUMPS logic | * Conduct analysis of any MUMPS logic currently used by the Allergy application * Add FileMan Entity file entries as required to support backward compatibility to FileMan data dictionaries and SSO security |
| Create a distinct VICS micro-service to replace the Patient Problem Data Entry logical paths currently provided by RPCs and MUMPS logic | * Conduct analysis of any MUMPS logic currently used by the Problem application * Add FileMan Entity file entries as required to support backward compatibility to FileMan data dictionaries and SSO security |

**2.2 Developing a Service Layer to Emulate CPRS RPCs for Select Data Read/Write Transactional Functions for Outpatient Pharmacy Computerized Order Entry (CPOE) (PWS 5.2.2)**

**Understanding.** Team ProSphere understands that the VistA Adaptive Maintenance project needs to migrate the Outpatient Pharmacy Computerized Order Entry functions of CPOE from VistA into a distinct VICS. Unlike PDE, CPOE involves CPRS as well as non-CPRS users that use a legacy roll and scroll interface built into VistA in conjunction with a 3rd party service providing drug checking functionality. This new VICS will be developed as a Node.JS micro- service that provides web-friendly interfaces for new clients while maintaining backward compatibility for CPRS, and addresses issues that arise with a centralized service including location, time management, and synchronization.

**Feasible Approach.** Our development team uses our highly feasible approach previously described, and further detailed below, to migrate the PDE functions. Our development team creates an initial design of the new REST service emulation layer designed to replace the existing CPRS RPC interfaces based on the requirements defined in our elaboration process. These web services consist of an n-tier architecture design that considers and addresses potential issues that arise with centralized services including location, time management, and synchronization, while ensuring continued functionality of CPRS and allowing for the retirement of the equivalent functions in the existing 131 VistA systems.

**Analysis of the Drug Checking Interface.** CPOE functions currently interact with the Drug Checking RPC interface called from VistA. Our RAs apply our Requirements Elaboration process (**Exhibit 3**) to identify all affected business processes utilizing this interface to provide a service emulation layer which provides forward and backward compatibility.

**Web Client Demonstration.** We use a web-client to demonstrate a capability that allows Pharmacists to manage outpatient medications and illustrates Pharmacist interactions. Our developers use HTML, Cascading Style Sheets (CSS), and JavaScript to develop the web-client connecting to the VICS services for data access, manipulation, and business logic execution.

**CPOE VICS Micro-Service.** Our development team uses Microsoft® Visual Studio to create all Node.JS REST components, and will Unit Test using Karma/Jasmine, or other TRM-approved framework. We leverage Team ProSphere's experience with development of Node.JS REST services that provides RESTful APIs for VA FileMan and eHMP to develop the required Node.JS component modifications **(see Exhibit 8)** and enhancements listed below, **Exhibit 12**.

# Exhibit 12: Node.JS Components

|  |  |
| --- | --- |
| **Requirement** | **Components/Enhancements** |
| Create a distinct VICS micro- service to replace the CPOE logical paths currently provided by RPCs and MUMPS logic | * Develop a data abstract layer containing ORMs representing each data value output from the current RPCs which provides common syntax for function calls [(post, get, put, delete) or (create, read, update, delete)]. * Develop a data persistence layer that will cache non-personally identifiable data and is accessible to Node.JS as needed to improve performance and reduce network load. * Develop a business logic layer containing libraries for each existing business logical path which matches the same output as the corresponding RPC. * Develop an interface that returns a MUMPS formatted String response for each existing business logical path containing identical data elements to those currently returned by the corresponding RPC. * Develop an interface that returns a JSON formatted response for each existing business logical path containing identical data elements to those currently returned by the corresponding RPC. * Functional Test the corresponding REST interface to validate forward compatibility. * Regression Test the corresponding function in CPRS to validate backward compatibility. * Compare outputs from Baseline (tests of current CPRS functions), Functional (testing of REST interfaces) and Regression (tests of CPRS after rerouting) tests for each corresponding function to ensure that comparable |
|  |  |

|  |  |
| --- | --- |
| **Requirement Components/Enhancements** | |
|  | functionalities have been produced. |

Our development team uses Embarcadero® RAD Studio™ to modify existing RPC Broker interfaces that utilize the newly developed REST interfaces instead of the MUMPS database. Our team Unit Tests these interfaces using DUnit. Our proposed Delphi Component modifications are listed in **Exhibit 13.**

# Exhibit 13: Delphi Components

|  |  |
| --- | --- |
| **Requirement** | **Component Modifications** |
| Create a distinct VICS micro-service to replace the CPOE logical paths currently provided by RPCs and MUMPS logic | * Modify each RPC Broker interface to perform an XMLHTTP call in order to consume the newly developed corresponding Vitals VICS REST interface.   + Ensure that changes are the minimal amount necessary to ensure proper connectivity to REST interface that passes same parameters currently passed to RPC and receives an identical output from the REST interface to that of the existing RPC.   + Ensure that authentication is updated to current SSO standards and passes sufficient user identification-related data to properly authenticate the user. |

Our development team uses InterSystems® Caché Studio to support the use of FileMan Data Dictionary files needed to ensure backward compatibility, and performs Unit Tests using VATS, the TRM-approved tool that Team ProSphere developed. **Exhibit 14** shows enhancements.

# Exhibit 14: MUMPS Component Modifications

|  |  |
| --- | --- |
| **Requirement** | **Component Modifications** |
| Create a distinct VICS micro-service to replace the CPOE logical paths currently provided by RPCs and MUMPS logic | * Conduct analysis of any MUMPS logic currently used by the CPOE application * Add Entity file entries as required to support backward compatibility to FileMan data dictionaries and SSO security |

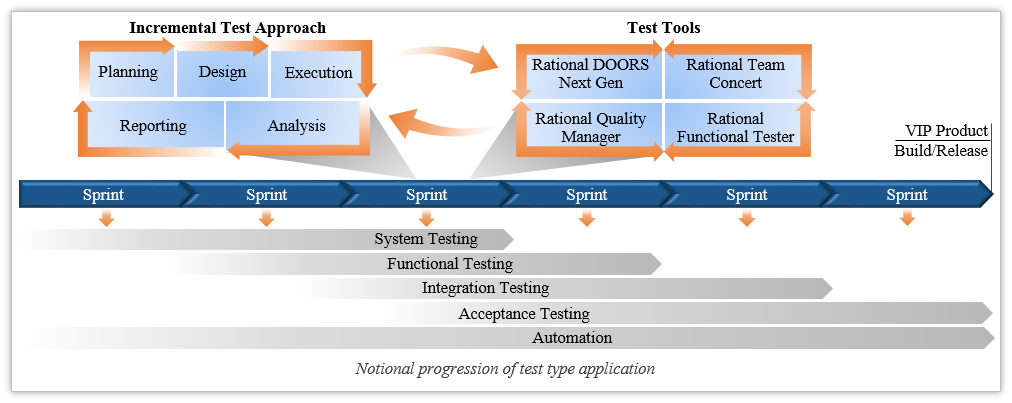
**2.3 Automated Testing of All Emulation, including Comprehensive Regression Test Suite (PWS 5.5.5, 5.5.5.1, 5.5.5.2)**

**Understanding.** We apply our extensive testing experience to leverage economies of scale, increased efficiencies, and improved reuse. Team ProSphere’s approach to testing for the VAM project focuses on creating efficient quality through consistent integration using TDD processes, integrated use with Rational tools, automated testing, and continuous integration testing techniques. Our Testers are fully engaged throughout the SDLC, and we matrix Testers in each Scrum Team. We also include Testers within our Scrum-of-Scrums so that they have a full awareness of the overall development status and how testing priorities may shift during each Sprint or throughout the Release. We coordinate with VA for all testing activities including identification, prioritization and tracking of requirements; software code and configuration changes; release and deployment schedule; integrity of single code base; and baseline configurations in all relevant environments. Our testing philosophy emphasizes automation, creating reusable artifacts, and integrating testing within our development Sprints to mitigate timeline, budget, and resource overhead and risk. We perform extensive testing throughout development and Initial Operating Capability (IOC) test events.

**Feasible Approach.** As shown in **Exhibit 15**, our incremental test approach relies on an iterative process through which we analyze, plan, design, execute, and report on each testing requirement. We align our approach activities with VIP processes. We communicate the start dates of testing three days prior to the beginning of each test so that VA can observe the testing aspreferred. We carefully assess each requirement, Epic and User Story, and work with stakeholders to ensure these items are captured and elaborated within the Rational tool (namely RDNG). At the start of the project, we establish the overall test strategy and Master Test Plan, and with each successive

build, we apply iterative test processes to the items in the Sprint backlog, maintaining alignment with the overall strategy and plan.

# Exhibit 15: Incremental Test Approach



This strategy includes non-functional requirements (e.g. load, performance, installation, back- out, and rollback). We test each sprint release and IOC site using each test event presented in **Exhibit 16**. We assist the VA in conducting the security scans, accessibility reviews, performance tests, technical standards review, architectural compliance assessments, user acceptance reviews and IOC tests, audits, and reviews.

# Exhibit 16: Testing Event

|  |  |
| --- | --- |
| **Testing Event** | **Description** |
| **Unit Testing** | Testing is a major focus in our development process. Our Testers are matrixed into Scrum Teams and work with Developers across Sprints to perform Unit Testing. We use TDD processes to focus on creating and executing Unit Tests so that defects are identified and resolved early and efficiently. These Unit Tests exercise code at the lowest level and in isolation of all other code to produce targeted test scenarios. We write Unit Tests prior to developing the actual software to ensure focus on the requirements before writing the code. A Developer begins the process by completing a Unit Test and then running it to ensure that it fails; since the test is written before the software, there is no way for it to succeed initially. Once it is proven that the initial test failed, our Developer refactors the software and finishes when all tests have passed. We use SonarQube to monitor Unit Tests, line, and code coverage. Our Testers use the Master Test Plan to determine what types of tests are to be automated. They write test scripts that are stored within Rational and become a part of a larger automation testing suite. |
| **Functional Testing** | Our testers use manual and automated testing that examines the front- and back-end functionality of the code to ensure that all the data entry, user interface, and application calculations are working correctly. Functional Testing of the release is done in such a way as to provide independence and provide a validation and verification that the testing results are not influenced by other groups. Tests and results can be influenced by the knowledge gained from other testing, from development teams, and business teams but will be based on business feature requirements that provide tests and results that are independent from other testing and any bias from other groups. This testing is performed to validate that the software satisfies all requirements in the current approved Release Cycle Scope, found in Rational Quality Manager. This phase is conducted with the primary goal to ensure that the CPRS components are performing as specified. Functional Testing occurs in accordance with the Master Test Plans. The Development Manager verifies that test cases are adequately  reviewed by the respective Sprint team to ensure consistency, understanding, and correct focus. The test cases must provide adequate tests to demonstrate that the software satisfies functional and accessibility requirements, as documented in the user stories and Section 508 respectively. Where possible, the Test Scripts will be automated using TRM-approved test automation software. Prior to entering a functional test cycle, we verify the input criteria have been met. The input criteria may include Unit Test results, a completed smoke test, and release notes or Version Description Document (VDD). When the test result of a test case deviates from the expected results, the actual results shall be documented. Software testing completes when the exit criteria are achieved. Exit criteria typically include what functionality was tested, the test results, and a list of any identified defects. |
| **Regression Testing** | As new builds are delivered, we conduct Regression Testing to confirm that changes and/or additions have not broken existing functionality. We use the existing test scripts and cases that were stored in the Rational repository. This approach identifies bugs that may have been accidentally introduced into a new build. We build Regression Test suites from these test cases and incorporate them into Regression Tests conducted in all environments. |

|  |  |
| --- | --- |
|  |  |
| **Testing Event** | **Description** |
| **Component**  **/System Testing** | Our Team provides Component/System Testing to expose interface defects and defects in the interaction between integrated components and to verify installation instructions. Our goal is to test and validate the system(s) for full-scale deployment. Component/System Testing takes place in parallel with the Agile development process, which includes activities such as requirements definition, software design, coding, source code control, code reviews, change management, CM, testing, release management, and product integration. We provide testing with the goal of testing and validating the system(s) for full-scale deployment. We take an Agile test-driven approach to system testing using a system, where testing cycles are predictable with defined functionalities. This philosophy reduces cost-of-delay, delivering quality, performance, and oversight for VA throughout the software development lifecycle. Necessary to achieve interoperability with various VA systems, we perform Integration Testing to verify that data exchanges between internal and external applications perform without delay and without any loss in functionality. Our team analyzes system requirements, data formats, and performs data validation to write test cases that allow us to test known dependencies between applications, and to identify unintended and unexpected dependencies. Whenever a new test case is written, we store it in RQM to be used in future integration tests, enabling us to leverage existing test cases when building new ones, accelerating test cycles throughout the project |
| **Performance Testing** | Performance Testing will be based on SLAs and available benchmark data identified in the BRD, RSD, and/or Software Design Document (SDDs) throughout the project lifecycle to determine if the speed, effectiveness, efficiency, reliability, usability, and interoperability of CPRS were compromised. Our testers will use the VA's standard approach and HP Performance Center (HP PC) as well as IEEE, FISMA, and National Institute of Standards and Technology (NIST) approved standards (as necessary) to execute Performance Test plans – created from the existing test scripts – and gather the required information for the Performance Test being executed. Our testers will simulate different simultaneous user |
|  |  |
|  |  |
|  | loads on the CPRS components as well as the integrated system. We will analyze the performance test results to document any performance effects, and where performance can be improved. The results of these tests will be disseminated to the development teams for knowledge and planning purposes to address any potential issues. If/when CPRS’ or one of its components’ performance is found to be inadequate, our testers will use Dell Foglight to monitor CPRS in real time and collect pertinent information. Dell Foglight gives our testers the ability to perform deep dive analysis on the Application Infrastructure, Transactional Activity at the application layer, as well as code level root-cause analysis. Based on our experience we are able to use Performance and Monitoring tools to ensure that performance problems cannot only be identified quickly but can be traced to upstream/downstream systems and triaged accordingly. Performance Testing will be executed and documented daily throughout the lifecycle of the project, and the final performance test results will be documented in the in the Post Development Test Report. |
|  |  |
| **Cybersecurity Vulnerability Scans and Remediation** | The VA Office of Information Security (OIS) Software Assurance (SwA) Program Office provides the scanning software (HP Fortify) to developers and test teams to perform static code analysis, providing users with immediate results regarding code vulnerabilities. We review the form and the responses to the request checklist to confirm the availability and hygiene of the source code, fortify scan results as conducted by the developer, and develop a summary description of the application architecture. We then perform and support both manual and automated static source code reviews and provide remediation guidance. Our team provides an independent assessment and verification that the scan provides 100% coverage of all custom VA application code. Our team substantiates the developers’ test results and coordinate validation of our findings with the SwA Program Office in support of these activities. Additionally, we perform all other required system and application level mandated security control testing and validation. Our testers work with the IA Team to schedule and run Fortify scans as a part of our overall testing examining IA concerns with regard to software operation and user interaction so any potential IA issues with regards to software functionality or a user using the software can be highlighted early. |

**Master Test Plan.** At the beginning of the contract, our Test Manager creates a Master Test Plan that explains our testing strategy, criteria, deliverables, schedule, environments and locations, risks, metrics, processes, roles and responsibilities, and the breakout of manual versus automated testing. The Master Test Plan (MTP) includes the methods by which expected results will be achieved, deliverable artifacts to tests to be performed, and how test results will be validated by the Government. These detailed test plans, procedures, and scripts to ensure all requirement elements are addressed. We provide the MTP to the COR for review, and only after approval is provided will we conduct testing. We store all artifacts within RQM, including both automated and manual test scripts and test cases for all Epics and User Stories, and all test results. With Rational, we will be able to generate reports with updated information in the RTM that indicate the degree of coverage in our testing, the number of identified and resolved defects, and the trend line for testing completion. We set up a real time view of the integrated RTM using the tools and dashboards within RDNG. This provides oversight and traceability to help identify scope creep and to ensure no requirement is overlooked.

**Test Cases/Scripts and Traceability.** Following our MTP that outlines our Automation Test Strategy and Automation Test Suite Details, we perform automated testing for Unit, Smoke, Installation, System, Regression, and Performance Testing. Where appropriate, we use HP UFT,

Rational Functional Tester (RFT), and other test automation tools to accelerate highly repetitive or laborious test sequences. We enter all the manual and automated Test Cases, Data, Scripts, and Results into RQM. We ensure these items are properly linked to work items in RTC as well as requirements stored in RDNG. After we have captured and linked the data, our RM uses the custom dashboard features of RDNG. As we have in FileMan and other CPRS projects, we create custom Widgets within RDNG to display real time information for the Test Evaluation Summary (TES), Agile Defect Log, Test Log, RTM, Environment Configuration, and Interface Specifications for each build on the dashboard. We create and share multiple custom dashboards for the project team, VA PM / COR, and business owners to ensure they are consistently able to view the information that is most important and useful for their decision making. We modify the data presented as needed to satisfy the Agile needs of the project. Throughout the testing process, we record test activity and test results, again leveraging Rational Tools, and updating the Test Results Report daily with the number of tests performed; the number of critical, high, moderate, and low defects documented; and any issues/concerns. These defects and issues are then added to the Sprint Backlog for resolution in the next development Sprint. Upon completion of all testing we provide the testing results in a TES, highlighting results to aid in a high-level assessment.

**Automated and Regression Testing.** Automated Functional and Regression Tests create repeatable, user-driven scenarios that simulate a user clicking-through the application without the need for an actual human to execute the clicks each time. This improves the number of test scenarios executed by reducing/eliminating the human intervention in each test case. By repeatedly running the same tests, our Testers can immediately identify broken scenarios. We always execute a limited number of targeted tests manually, but rely heavily on our test automation suites to account for full breadth and depth of tests that must execute CPRS functionality. We also conduct extensive testing related to non-functional requirements, (e.g., load, performance, installation, back-out, and rollback).

Our Teams create Functional, Integration, and Regression Test scripts to assure 90% total coverage with test cases. We develop software test documentation as required throughout the release cycle to include test case documents, test plan, testing strategy, testing scripts, test data, traceability data and artifacts, test results, and defect reports. We provide a Testing Summary Report at each Sprint Review throughout the release cycle, and a Final Test Report at the end of the release cycle which documents the results of testing. We identify the test inputs necessary to stimulate the product components and the outputs that measure system responses and proper functionalities. Test cases are developed to establish clear, precise understanding for external test teams with concise steps. Our team prioritizes higher risk functionality criteria for the development of test cases to demonstrate that requirements are met and to illustrate risk-based testing. Below are contributors in determining risks that we account for in test case prioritization:

* **New Features:** Features with greatest uncertainty will be a top priority for testing, this includes performance;
* **Interoperability:** Compatibility with external systems and non-standard communication protocols are common, essential risk points; and
* **User Interfaces:** How well application users can control operations that is, the effectiveness of the system human-machine interfaces, must be determined.

As we currently do on FileMan23, we maintain the existing automated VistA patching process for the environments for continuous patching of the environments to ensure software is always

compatible with updates made to the production environment. We update the patch checklist utility application to automate the steps required to complete the patch checklist, **Exhibit 17**.

# Exhibit 17: Patch Checklist

|  |  |
| --- | --- |
| **Patch Checklist** | |
| Populating lookup tables | VistA account lookups |
| Adding a patch description | VistA package lookups |
| Uploading captures | VistA package file lookups |
| VistA package mapping lookups | VistA integration agreement lookups |
| Using data from a previous build for evaluation of related checklist steps | Checklist and checklist detail screens to view and update checklist steps |
| Completing checklists, to view automated and manual results, update step results and generate reports | Automatically updating any relevant automated patch checklist step, when uploading captures |

Successful automated testing solutions drive operational productivity, increase code coverage, promote software quality, and allow our testing resources to accomplish more, better handle surges, and reuse test cases and test scripts in a highly efficient manner. These test scripts are run in the Test Environment every three hours, and reports sent to the PM, Scrum Masters, and Tech Lead when defects are identified. Each night test scripts are run again in conjunction with Sonar to determine daily code quality and test coverage. During each sprint we run automated scripts and Regression Testing to ensure that the Burndown Chart shows no open defects. As seen in **Exhibit 18**, our approach incorporates existing VA processes and tools including the Rational tool set.

# Exhibit 18: Team ProSphere’s Automated Testing Approach



We use the TRM-approved ***VistA Automated Test Suite (VATS)*** that Team ProSphere created for FileMan 23 to automate the Unit, Functional, Component Integration and Regression testing of all MUMPS code, and web services making use of the cURL command capabilities in VistA. This tool allows us to create test scripts that we store in RQM and then run automatically to perform the same automated testing above on MUMPS code allowing a time and cost savings compared to doing it manually. VATS performs automated testing in any VistA environment mimicking (not replicating) commands executed in Attachmate Reflections. The system is fully extensible, configurable and available for Unit, Component Integration and System, Functional and Automated Regression. VATS is fully VIP compliant and all scripts and results automatically import to RQM.

**Formal Testing Results.** We document all test results, defects, and interface interaction issues in the Development Testing Report that includes the status of all testing of Post Development Test Scripts, configurations, utilities, tools, plans results and all relevant metrics. It is important to verify installation instructions are correct and associated processes run smoothly. Results are recorded in the RQM and stored for future use. We manage, track and remediate findings and defects from all associated tests before submitting the release for approval. Upon completion of the VIP-driven Agile development phase of testing, we coordinate with the COR to gain appropriate access to the government-specific testing environment to perform the Final Test.

Before moving on to IOC we perform demonstrations to the VA COR and stakeholders that new interfaces perform as designed in accordance with system requirements.

**2.4 Initial Operating Capability (IOC) Support Demonstrating CPRS Retains Full Functionality against a Single Centralized Services Replacing those Functions of the Original, De-Centralized VistA Source Instances (PWS 5.2 and 5.6)**

**Understanding.** Efficient delivery of business value to VistA end users requires planning and execution of IOC in a way that maximizes the ability to identify defects and issues while minimizing schedule delays and other adverse events. IOC is a particularly high-risk phase of the overall VIP process given the complex technical and administrative tasks required to coordinate among the various IOC sites’ stakeholders and facility-level leadership. We are confident that Team ProSphere brings the experienced, seasoned professionals the VA VAM Team needs to keep the schedule on track while identifying and remediating any software defects or issues.

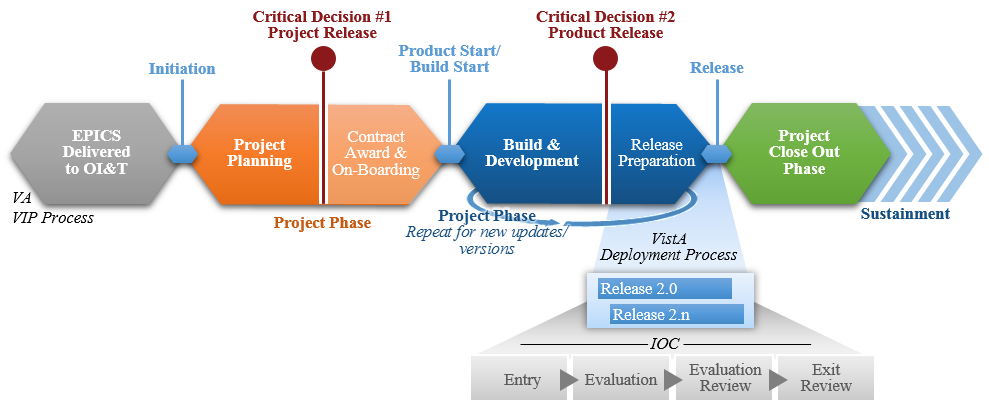
**Feasible Approach.** We are aware of the complexity of IOC site participation. For this reason, we seek to engage with the IOC candidates as quickly as possible for each build, and well before the Team reaches Critical Decision 2 (CD2) in VIP, as administrative delays at the site level can occur, but should not delay the overall project schedule. To support IOC, our Team collaborates with VA to verify the VA facilities enlisted to participate from the list as part of CD1. For an application of the size and complexity of CPRS, our Team recommends a minimum of 2 sites and maximum of 5 sites for participation with VHA end users. VHA participant Teams may be at the same or different facilities. Based on our past experience in performing IOC’s for VistA, CPRS and FileMan, selected sites should represent the entire spectrum of diversity across VA service areas with respect to size (small, medium, and large Veteran populations serviced), physical location (not all in the same VISN, ideally distributed across the nation), and type of population serviced (rural vs. urban). If sites have not been chosen or decline after CD1 then, working with the VA Network of Peers, we identify up to five IOC VA facilities: with two large, two integrated, and one medium/small VA Medical Centers (VAMCs) spread over multiple regions desired. After the list of participating sites is confirmed, we initiate the process of establishing MOU with the appropriate site leadership including the facility CIO, their delegates, and representatives. These, steps are likely on or near the project schedule’s critical path. Once the CPRS build has been verified to meet the Checkpoint Requirements **Exhibit 19**, our Team conducts an evaluation deployment to monitor production readiness prior to system deployment.

# Exhibit 19: IOC Checkpoint Requirements

|  |  |  |
| --- | --- | --- |
| **Checkpoint Requirements** | | |
| Traceability / Test Execution / Test Results / Defect Log | Documentation Updates to Anonymous Directory | Production Operations Manual/Technical Manual with RACI |
| Version Description Document | Risks | ATO (Approved) |
| 508 Compliance | User Guides | VDL Document Updates |
| IOC Test Plan and Artifacts | Deployment/ Installation/ Backout/Rollback |  |

This deployment as part of Initial Operational Capability (IOC) incorporates both an additional level of testing and the preparation and finalization of technical documentation and training materials. Lessons learned from this implementation will be taken into account for future CPRS release efforts. Team ProSphere recognizes a key goal of VIP is providing VA with a single, integrated release process focused on Agile processes and frequent releases, rather than on documentation. Additionally, it provides critical enterprise-level tools and recommends Agile approaches at all levels. Our VIP-centric IOC approach allows multiple releases to be going through IOC without interfering with each other or the normal activities of a VA site while still fitting into the process, as illustrated in **Exhibit 20**.

# Exhibit 20: How IOC Fits into the VIP Process



IOC is broken into 4 distinct phases, **Exhibit 21** outlines these phases and our approach.

# Exhibit 21: IOC Phases and Implementation

|  |  |
| --- | --- |
| **IOC Phase Implementation** | |
| **IOC Entry** | The objective of this activity is to obtain approval for the request to install the Release code into the live production environments at selected field sites for execution of the IOC Evaluation from the IOC Review Panel. As input for this activity, the participants use the ESE Enterprise Testing Findings Report, the ESE Testing Findings Report, the IOC Site Memorandum of Understanding, the Master Test Plan for each application, the National Release Checklist, the Release Notes, the Risk Management Plan, the System Quality Assurance Findings Report, the completed Test Cases & Scripts, and the Updated Patch or New Patch documentation to determine whether the application is ready to proceed to IOC Evaluation.  The Test Team works with the VA PM to complete the IOC Request forms (the IOC Record of Request, the IOC Entry Request and Exit Summary, and the IOC Release Authorization Approval). The Project Manager, in coordination with the IOC Release Coordinator, conducts the IOC entry meeting (Evaluation Site Kick-off) with all IOC stakeholders to determine whether to proceed to the IOC Evaluation activity and obtain the approval of the IOC Go/No-Go Decision Memorandum.  During the IOC Entry call, a presentation of event expectations, communication methods, and testing processes are presented to the IOC reviewers. This call offers the field sites an opportunity to ask questions and validate processes. |
| **IOC**  **Evaluation** | We perform an IOC evaluation event of the release to the concurrent VA facilities. We run all of the automated tests that have been created and verified as part of the process above except Unit Testing (Section 2.3) on the IOC sites systems. We use these tests to determine if any of the custom configurations or Class 3 code affects the updates. We then work with those sites to address any custom configuration issues that might are identified. We work with the IOC sites in situations where site customizations cannot be restored. Our team provides all testing, deployment and QA documents to the VA release agent at the beginning of the evaluation testing phase. We provide any assistance or updates to those documents as requested.  Team ProSphere communicates with the VA PM regarding all updates and changes to the Release and development schedule. We also work with all test sites during pre-deployment IOC. This ensures that every site has the capability and training to immediately use the software solution after we have deployed or updated the system. To comply with contingency planning per the Federal Information Security Management Act of 2002 (FISMA), Team ProSphere, as part of the limited deployment, conducts a test of the Implementation and Deployment Plan. The test of the contingency plan involves using system backups to recreate a working system in a virtual environment to validate viability. In the Deployment Plan, Team ProSphere includes steps to do a full or partial rollback of the deployment should it be necessary. We design this system rollback to occur with little to no impact on the Medical Center and system data.  We provide a lessons learned document after the successful completion of IOC evaluation. We use these lessons learned to update the Deployment Plan, an essential part of the Enterprise Implementation Plan. At the end of the Phase 3 a completed draft of the implementation plan will be available for VA review and approval. Our team then moves forward with full deployment according to the approved Implementation plan. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
| **IOC Phase** | **I** |  | **mplementation** |
|  |  |  |  |
| **IOC**  **Evaluation Review** | The objective of this evaluation activity is to validate that each production Release has been installed correctly into the live production environment, and that the Release features are functioning as the business requirements specified. Additionally, this test event validates that the Release does not adversely affect the existing functionality of the current product/systems.  As input for this review, the team uses the IOC Site Memorandum of Understanding for the application, the IOC Site Concurrence Statement, the IOC Site Evaluation Log, the IOC Site Evaluation Defect Log, the IOC Entry Request and Exit Summary, the Master Test Plan for the application, the associated Test Cases & Scripts, the Health Product Support Approval Document, the Lesson Learned Report and the Registered Request for Change. With the participating site list baselined for each entry to IOC, all other IOC entry criteria confirmed including the VIP CD 2 ruling, testers secured for each site, and confirmation to proceed granted by the VHA System Implementation Office, we initiate IOC. The IOC testingop rhythm is as follows:  Testers follow clearly documented test plans that include instructions for how to record defects they identify, and with whom to follow up if they have questions on how to follow the plan or any related matters. We provide site participants with a template that guides them on how best to define the issue they have found, how the development team can reproduce the issue, and any other details that may be pertinent. Testers submit findings to a single, clearly identified Point of Contact (POC)  The POC analyzes defects reported from the field to gain an understanding of what has been reported. The POC then adds the defect(s) to a list of IOC defect log for discussion. If there is any ambiguity in the defect as it has been reported, the POC calls the field tester to discuss, and add further detail to the finding.  **The Team meets daily during the IOC phase to:**  Rapidly gain a shared understanding of any identified defects, consulting with the intake POC to verify their understanding is correct and complete; Devise the most efficient and technically sound method to resolve; Estimate the relative “size” of the resolution; Consolidate all data gathered in the previous steps into Rational which includes the development teams’ anticipated schedule impact and timeline for resolution, along with any additional impacts to the software code base and documentation.  Upon COR approval of the plan, add the defect(s) to the overall product backlog, and execute resolution(s), testing of the resolution, and release of the resolution(s) as required. Each defect resolution or package of resolutions triggers updates to the Software Source Code, Compiled Code and Supporting Documentation.  Review the statuses of any other open defects previously identified and discuss any schedule issues, potential risk, issues, or blockers related to them. Review the IOC Defect Tracking Spreadsheet to facilitate the DMB for adjudication and concurrence from all DMB members.  This is an iterative process that continues throughout IOC. Our Team collaborates with the COR and other stakeholders to categorize defects by severity, and prioritize them for resolution accordingly. “Critical” defects or “showstoppers,” those defects identified that make the application or an important feature otherwise unusable with no available workaround, are | |
|  |  |  |  |
| categorized for resolution first, with no other defects considered for resolution until they are resolved. We prioritize “High” and “Medium” defects next. “Low” categorized defects are generally considered low priority for resolution, and may be acceptable to keep unresolved in production depending on both input from and negotiation with the COR and the capacity of the development team at any given time. As our Team of developers resolves the defects in accordance with our rhythm for identifying, analyzing, resolving, and re-testing features, we coordinate the installation of the software update into the test site production accounts, and deliver updated documentation, wherever possible leveraging our data stored in Rational to reduce risks to version control and support the ability for the entire Team to see the updates in Rational as they happen.  This phase in the process requires 10 days of error free IOC testing. After the Team has completed code changes and the COR has given approval, IOC continues with the newly baselined code for an additional five days (unless shortened according to the preferences of the COR and government business representative) to verify that the code and overall technical environments are stable and ready for wider release to the field.  At the end of the IOC Evaluation event, the IOC Release Coordinator obtains the IOC Site Concurrence Statements document and the IOC Site Evaluation Log. The VA PM reviews the findings from the IOC Evaluation and update the IOC Entry and Exit Summary document, and the IOC Exit Summary in the Issues, Anomalies, and Exceptions section and in the Risks and Mitigation Strategy section. The VA PM and the IOC Implementation Manager reviews the results from the IOC sites and Health Product Support Review to determine whether to proceed into full deployment and the VA PM and IOC Release Coordinator creates a Lessons Learned Report. The IOC evaluation will not be completed until all planned production releases have finished their individual IOC evaluations and have been accepted by the field test sites.  We work closely with the external test teams to show that any failure and defect results are clear. Our team monitors and schedules progress throughout testing activities, and verify test completion rates against the execution metric expectations as predefined in the test plan. Any deviations are identified and addressed upon detection. All defect discoveries are documented, and reports generated, as defined in the Test Plan. Once testing is complete, all results are collected and consolidated into the formal template. We report test results to the identified stakeholders according to the process identified in the Test Plan. The Bi-weekly Status Report summarizes all testing activities, with emphasis on results, team performance, and adherence to the project schedule. | | |
| **IOC Exit** The objective of this activity is to obtain approval of the Initial Operating Capability Entry Request and Exit Summary, to  **Review** obtain a successful IOC sign-off and to achieve a recommendation for National Release from the selected IOC Evaluation field sites. The IOC Exit Review participants include the VA PM, IOC Stakeholders, Service Delivery and Engineering Field Operations, Field Operations and Development Implementation Manager, Release Manager, Health Product Support and other stakeholders as identified. Our approach ensures a tested and documented National Release.  The IOC Release Coordinator obtains the Site Concurrence Statement and sign-off. The Pass Code and Transition Document are provided to Product Development Support for review and approval. Then the IOC Exit review, which includes the Health Product Development review, is held. The review meeting activities include but are not limited to the VA PM and the IOC Release Coordinator presenting their analysis of the IOC Evaluation to all IOC stakeholders and a determination of how to | | |
|  | | |

|  |  |
| --- | --- |
| **IOC Phase Implementation** | |
|  | proceed. If all prerequisites are met and all approvals are received, this event leads to receipt of a recommendation for National Release and site approval.  The development of both the Implementation Plan and Release Checklist will be accomplished using the VA OI&T guidelines and jointly reviewed by SMEs and approved prior to release scheduled in POLARIS to the field for testing. We seek input on the implementation process from the Regional CIO(s) and facility POCs to maintain daily facility operations without interruption to the patient work flow at each IOC test site. We deliver project reviews that provide status of ongoing activities and planned activities for next deployment, and feedback into future deployments from lessons learned. Documentation of each stage of deployment will be maintained and shared with the team members at our weekly status calls.  Since the clinical community will be responsible for the national review and limited testing, we welcome their continued participation and input from the assigned POC/clinical team during the release and documentation stages so that a quality product is delivered and customer satisfaction is at the highest level before national release. We use motivation and communication techniques to improve clinicians’ and testers’ experience during IOC. We deliver project reviews that provide status of ongoing activities and planned activities for next deployment, and feedback into future deployments from lessons learned. Patch Release Documentation and step by step installation guides will be produced that are Section 508 compliant and which follow the guidelines listed in the RSD. Checkpoints are established to ensure that accessibility is incorporated from the earliest possible design or acquisition phase and successfully implemented throughout the project. The product will be made available in RTC and the VDL document library. |
| **2.5 National Deployment Following IOC so CPRS Retains Full Functionality against a Single Centralized Service Replacing the Functions of the Original 131 De- Centralized VistA Source Instances (PWS 5.7)** | |

**Understanding.** Upon the successful completion of IOC, the VA requires that the newly created VICS centralized service be released through a National Deployment. Once the centralized service is deployed, the VA intends to begin the process of retiring the 131 de-centralized VistA instances while still maintaining full continuity of the service in the CPRS client. Team ProSphere understands that as a part of the national deployment and retiring process, defect management and other technical support will be required for both the newly created central services and the programs being retired. Our team uses its highly feasible approach described below to provide these technical services and capabilities.

**Feasible Approach.** We utilize the experienced gained from successfully supporting multiple national releases across several VA projects to provide a full complement of Deployment Services. Our Deployment Services consist of addressing facility outreach, end-user training and acceptance and deployment planning allowing for the retirement of the specified functions of CPRS Patient Data Entry and Pharmacy Computerized Physician Order Entry (CPOE) products for installation into a VA-provided, FedRAMP approved, industry-standard, commercial cloud- based production environment. Our deployment process promotes a culture to collaborate with, problem solve, trouble shoot, and provide technical support to affected VistA project, Software Quality Assurance (SQA) and release managers. Additionally, during the deployment, we ensure no loss of functionality through frequent communication by phone, mail and instant messaging with the individual Medical Centers. Due to the complexity of migrating existing logic (local instance MUMPS logic) into a new platform (Cloud-based Node.JS), we ensure proper deployment, configuration, activation, and validation of the newly developed VICS services prior to retirement of any existing MUMPS logic. The key steps are identified in **Exhibit 22**.

# Exhibit 22: Deployment and Retirement Tasks

|  |  |
| --- | --- |
| **Task Description** | |
| **Deployment** | Setup and configure a Cloud Service, as defined by VA, which may be VA provided or commercial service, such as Amazon Web Services (AWS) or Microsoft Azure, for deploying the VICS services.   * Deploy Common VICS Micro-Services onto Cloud Service * Deploy PDE Vitals VICS Micro-Service onto Cloud Service * Deploy PDE Allergy VICS Micro-Service onto Cloud Service * Deploy PDE Problem VICS Micro-Service onto Cloud Service * Deploy CPOE VICS Micro-Service onto Cloud Service |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| **Task** |  |  |  |  |  |  | **Description** |
|  |  |  |  |  |  |  |  |
| **Configuration** | Configuration of services includes database connect   * Configure Common VICS Micro-Services * Configure PDE Vitals VICS Micro-Service * Configure PDE Allergy VICS Micro-Service * Configure PDE Problem VICS Micro-Service * Configure CPOE VICS Micro-Service |  |  |  | i |  | ons, service connections, and other standard web service related logic. |
| **Activation** | Activation of services includes setting up operating s start/stop/restart during scheduled/unscheduled serve   * Activate the Common VICS Micro-Services * Activate the PDE Vitals VICS Micro-Service * Activate the PDE Allergy VICS Micro-Servic * Activate the PDE Problem VICS Micro-Servi * Activate the CPOE VICS Micro-Service | |  | ystem level service for each VICS so that the service will r shut downs.  e ce | |
| **Validation** | Validation of services includes performing all steps |  |  |  | defined in the IOC Test Plan for each VICS, with special attention to | |
|  |  |  |  |  |  |  |  |
| performing Comparative and Regression Testing. Comparative Testing will require that the same tests be conducted against both existing (non-modified) system and the new VICS based system to ensure all functions operate identically. Regression Testing will cover all areas in CPRS that share common coding logic or RPCs in particular for the Patient related data RPCs.   * Validate the Common VICS Micro-Services (these systems are validated passively and concurrent with the testing of the application VICS) * Validate the PDE Vitals VICS Micro-Service * Validate the PDE Allergy VICS Micro-Service * Validate the PDE Problem VICS Micro-Service * Validate the CPOE VICS Micro-Service | | | |
| **Retirement** Retirement of the Patient Data Entry services existing logic can only be done once all new VICS logic and services have been validated, and only for those RPCs and logic which are not used by any other parts of CPRS unless those areas use identical data content, structure and format. Any deviation or variance to the output may require that those RPCs be left in place to support those areas which are outside the scope of this project. These areas will be identified during the Analysis Phase and verified to have impact during Functional/Regression Testing prior to IOC. Retirement process will be done in a single enhancement patch to the RPC Broker which will reroute existing calls to the new service emulation layer (VICS). | | | |

In support of these deployment tasks, our RM utilizes POLARIS 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. For each release, they provide data (IOC entry, exit, & production dates, description, dependencies and impacts, downtime or unavailability including duration, post activity validation) for populating and updating the POLARIS calendaring process for each release and deployment. Our Security Analyst, in conjunction with our development team, collaborates with the EPMO team to provide the necessary data and information to support ATO approval. We provide implementation support for all development projects in accordance with VIP and PAL. Our IOC test plans provide for backup and recovery contingencies during a release of software that are extended to National Release. We update and provide all required documentation to VA staff involved in the release including but not limited release managers/coordinators, activity coordinators and quality assurance staff. We update as necessary the Production Operations Manual (POM)/Technical Manual (VistA) with “Responsible, Accountable, Consulted, Informed (RACI)” information as well as User Manuals. Likewise, the Deployment, Installation, Back-out, Rollback Plans will be revised as need based on feedback from the field.

We support operational readiness and transition planning, IOC and national deployment release coordination, OI&T project documentation reviews, roll-back planning and mitigation, and go- live coordination. Our approach includes our RM coordinating test site calls with VA staff, VA Release Managers, and VIP Release Agents to ensure that all deployment activities are closely coordinated with the physical release of software; maintaining pre-deployment checklists to ensure all deployment and training activities are completed on schedule. We schedule calls with the Release Managers and the Medical Centers to insure the staff is prepared to support the testing and deployment activities. Once each deployment is complete, we conduct lessons

learned meetings during and after implementation as defined by VIP/PAL. During the 90-day post deployment support, should a defect be identified by the Government and its designees, we use our defect management process to triage, perform root cause analysis to determine a resolution, and then create and implement a remediation plan to address the defect. The remediation plan contains the plan for resolution, timelines associated, and document update.

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

**Exhibit 23** summarizes the Level of Effort associated for PWS sections 5.2, 5.3, 5.5, 5.6, and

5.7, and all subparagraphs (5.2.1, 5.2.2, 5.5.1-5.5.6, and 5.7.1). All work performed under this TO will be on a firm-fixed price basis at our Alexandria VA and San Diego CA offices. To determine this contract’s required staffing and LOE, we considered the following: activities to be performed in support of the PWS requirements; skills required to complete each activity; schedule milestones, task dependencies; likely areas of complexity and potential re-work; and our previous experience executing similar projects.

# Exhibit 23: Level of Effort

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task** | **Labor Category** | **Role** | **Base** | **OP 1** | **OT 1** | **Company** |
| 5.1 | Project Manager | Project Manager | 960 | 960 |  | ProSphere |
|  | Configuration Manager | Release and Config Manager | 960 | 960 |  | ProSphere |
| 5.2 | Development Manager | Development Lead | 480 | 480 |  | ProSphere |
|  | Security Analyst | Security and A&A support | 960 | 960 |  | ProSphere |
|  | Project Manager | Technical Project Manager | 960 | 960 |  | ProSphere |
| 5.2.1 | Software Engineer Sr. | Technical Lead | 1920 | 1920 |  | ProSphere |
|  | Developer Expert | MUMPS Developer | 1920 | 1920 |  | Medsphere |
|  | Developer Sr. | CPRS/Node js Developer | 3840 | 3840 |  | ProSphere |
|  | Technical Writer | Technical Writer | 480 | 480 |  | ProSphere |
|  | Functional Analyst | Requirements/Build Analyst | 960 | 960 |  | TSC |
| 5.2.2 | Software Engineer Sr. | Technical Lead | 1920 | 1920 |  | Cognitive |
|  | Developer Expert | MUMPS Developer | 1920 | 1920 |  | Medsphere |
|  | Developer Sr. | CPRS/Node js Developer | 3840 | 3840 |  | Cognitive |
|  | Technical Writer | Technical Writer | 480 | 480 |  | ProSphere |
|  | Functional Analyst | Requirements/Build Analyst | 960 | 960 |  | TSC |
| 5.3 | Software Engineer Sr. | Technical Lead | 1920 | 1440 |  | ProSphere |
|  | Test Engineer Sr. | Tester | 960 | 720 |  | ProSphere |
|  | Developer Sr. | Sr. Developer | 1920 | 1440 |  | ProSphere |
| 5.4 | Functional Analyst | Requirements/Build Analyst | 960 | 960 |  | ProSphere |
| 5.5 | Development Manager | Development Lead | 480 | 480 |  | ProSphere |
| 5.5.1 | Development Manager | Development Lead | 960 | 960 |  | ProSphere |
| 5.5.2 | System Engineer Sr. | Scrum Master | 960 | 960 |  | ProSphere |
| 5.5.3 | System Engineer Sr. | Scrum Master | 960 | 960 |  | ProSphere |
|  | Configuration Manager | Release and Config Manager | 960 | 960 |  | ProSphere |
| 5.5.4 | System Administrator Sr. | System Admin | 1920 | 1920 |  | ProSphere |
| 5.5.5 | Test Manager | Testing Lead | 960 | 960 |  | ProSphere |
|  | Test Engineer Sr. | Tester | 1920 | 1920 |  | Medsphere |
|  | Test Engineer Sr. | Tester | 1920 | 1920 |  | Cognitive |
| 5.5.6 | Security Analyst | Security and A&A support | 960 | 960 |  | ProSphere |
| 5.6 | Test Manager | Testing Lead | 960 | 960 |  | ProSphere |
|  | Test Engineer | Tester | 960 | 960 |  | ProSphere |
| 5.7 | Deployment Manager Sr. | Deployment Manager |  | 1920 |  | ProSphere |
| 5.7.1 | Software Engineer Sr. | Technical Lead |  | 480 |  | ProSphere |
|  | Test Engineer Sr. | Tester |  | 240 |  | ProSphere |
|  | Developer Sr. | Sr. Developer |  | 480 |  | ProSphere |
| 5.8 | Project Manager | Technical Project Manager |  |  | 40 | ProSphere |
| **TOTAL** | | | **42240** | **44160** | **40** |  |