

Patient-Centered Management Module Redesign (PCMMR)

Document Version 1.9

System Design Document



June, 2014

Revision History

Date	Version	Description	Author
12/16/2011	0.2	Initial draft (Information tranfered from SAD)	
2/27/2012	0.3	Updates	
2/29/2012	0.4	Technical Edit	
4/2/2012	0.5	Updates	
4/6/2012	0.6	Technical Edit	
6/4/2012	0.7	Updates	
6/4/2012	0.8	Technical Edit	
6/15/2012	0.9	Updates	
6/18/12	0.95	Technical Edit	
7/9/12	1.0	Incremented document version to 1.0 and added signature page	
7/30/12	1.1	Incorporated feedback from VA	
11/13/12	1.2	Added MVI interface details	
1/3/13	1.3	Updated with feedback from MVI	
1/8/13	1.4	Technical Edit	
1/21/14	1.5	Content Revisions	
2/19/14	1.6	Merged with Init7_PCMMR_SDD.docx	
4/14/14	1.7	Updates for MVI section	
6/11/14	1.8	Updates	
6/12/14	1.9	Updates	

Table of Contents

1. Introduction	1
1.1. Purpose of this document.....	1
1.2. Identification	2
1.3. Scope.....	2
1.4. Relationship to Other Plans.....	2
1.5. Methodology, Tools, and Techniques.....	2
1.6. Policies, Directives, and Procedures	4
1.7. Constraints.....	4
1.7.1. TRM.....	5
1.7.2. Release Architecture	5
1.8. Design Trade-offs	6
1.9. User Characteristics	7
1.9.1. User Problem Statement	7
1.9.2. User Objectives.....	7
2. Background	7
2.1. Overview of the System	8
2.2. Overview of the Business Process	10
2.3. Business Benefits.....	18
2.4. Assumptions and Constraints.....	19
2.4.1. Design Assumptions	19
2.4.2. Design Constraints.....	19
2.5. Overview of the Significant Requirements	20
2.5.1. Overview of Significant Functional Requirements	20
2.5.2. Functional Workload and Functional Performance Requirements ...	21
2.5.3. Operational Requirements	23
2.5.4. Overview of the Technical Requirements.....	23
2.5.5. Overview of the Security or Privacy Requirements.....	27
2.5.6. System Criticality and High Availability Requirements.....	28
2.6. Legacy System Retirement	28
3. Conceptual Design.....	29
3.1. Conceptual Application Design	29
3.1.1. Application Context.....	29
3.1.2. High Level Application Design	31
3.1.3. Application Locations	33
3.1.4. Application Users	33
3.2. Conceptual Data Design.....	34
3.2.1. Project Conceptual Data Model	34

3.2.2.	Database Information	34
3.3.	CISS Infrastructureincluding OHRS and PCMMR partner applications	35
3.3.1.	System Criticality and High Availability.....	37
3.3.2.	Special Technology	38
3.3.3.	Technology Locations.....	38
3.3.4.	Conceptual Infrastructure Diagram.....	40
4.	System Architecture	40
4.1.	Hardware Architecture	40
4.2.	Software Architecture.....	41
4.2.1.	CISS	41
4.2.2.	Partner Applications.....	41
4.2.3.	Code Framework.....	41
4.3.	Software Architecture – PCMMR	41
4.3.1.	CISS Compared and Contrasted	42
4.3.2.	Standard Partner-System Implementation	42
4.3.3.	AJAX	45
4.4.	Communications Architecture.....	45
5.	Data Design	46
5.1.	Database Management System Files	47
5.1.1.	JPA mapping example.....	49
6.	Detailed Design	51
6.1.	Software Detailed Design.....	51
6.1.1.	Module “CRUD editors”	51
6.1.1.1.	Processing	52
6.1.2.	Module “Search Screens”	53
6.1.2.1.	Processing	54
6.1.3.	Module “AJAX methods & Dialogs”	55
6.1.3.1.	Processing	56
6.1.4.	Reporting.....	57
6.2.	Batch Processes	57
6.2.1.	Transfer Patients to Entire Team.....	57
6.2.2.	Patient Bulk Operations	58
6.2.3.	Patient Auto-Inactivation.....	Error! Bookmark not defined.
6.2.4.	Cluster Considerations	58
6.3.	Scheduled Jobs	59
6.3.1.	MVIPatientInboundProcessorJob.....	59
6.3.2.	MVIPrimaryViewAndRegistrationScheduledJob.....	59
6.3.3.	PatientAutoInactivationScheduledJob	Error! Bookmark not defined.
6.3.4.	PurgeCompletedReportsJob	Error! Bookmark not defined.

6.3.5.	PurgeJobExecutionResultsJob	59
6.3.6.	TeamValidationJob	59
6.3.7.	VistaCleanupJob	60
6.4.	Performance Enhancements	65
6.4.1.	Hibernate level 2 cache - ehcache	65
6.4.2.	JPA batch	65
6.4.3.	JPA fetch & QueryCustomization	66
6.4.4.	DB indexes	66
6.4.5.	Multiple unattended servers	66
6.4.6.	Distributed queues	Error! Bookmark not defined.
6.4.7.	Spring @Async / TaskExecutors	66
7.	External Interface Design	67
7.1.	Interface Architecture	67
7.2.	Interface Detailed Design	67
7.2.1.	Vista	68
7.2.2.	MVI	68
7.2.3.	CPRS	71
8.	Human-Machine Interface	72
8.1.	Interface Design Rules	72
8.2.	Inputs	72
8.3.	Outputs	75
8.4.	Navigation Hierarchy	79
8.4.1.	“Profile” example screen	80
8.4.2.	“List” example screen	82
8.4.3.	“Alerts” functionality	83
8.4.4.	“Context-Sensitive Help” functionality	85
9.	System Integrity Controls	85
10.	Appendix A	86
10.1.	Requirements Traceability Matrix	86
10.2.	Packaging and Installation	86
10.3.	MVI use case “Manage a Patient Panel Request” requirements	86
10.4.	Design Metrics	88
10.5.	Glossary of Terms	Error! Bookmark not defined.
10.6.	Required Technical Documents	88
	Attachment A - Approval Signatures	90

1. Introduction

The mission of the Department of Veterans Affairs (VA), Office of Information and Technology (OIT), Veteran Health Administration (VHA) is to provide benefits and services to veterans of the United States. In meeting these goals, OIT strives to provide high-quality, effective, and efficient Information Technology (IT) services to those responsible for providing care to the Veterans at the point-of-care as well as throughout all the points of the Veterans' health care in an effective, timely, and compassionate manner. The VA depends on Information Management/Information Technology (IM/IT) systems to meet mission goals.

Over time, the VHA has developed a Primary Care (PC) system that balances productivity with quality, access, and patient service. Management of patient panels in PC through mandatory and consistent use of the Primary Care Management Module (PCMM) has supported this system redesign. In a PC setting and in the Patient-Aligned Care Team (PACT) model, patients are assigned a Primary Care Provider (PCP) who is responsible for delivering essential health care, coordinating all health care services, and serving as the point of access for VA care. The PCP works together with a team of professionals which includes nurses, pharmacists, social workers, health care professions trainees, clerks, etc.

The PCMM software is considered to be an important component in measuring patient demand and PCP capacity to meet that demand, as well as reducing wait times. It allows users to set up and define treatment teams, assign positions to the team, assign staff to positions, assign patients to the team, and assign patients to a PCP. PCMM was developed to assist VA facilities in implementing PC. PCMM supports both PC and non-PC teams. Teams are groups of staff members organized for a certain purpose.

In order to fully support a team-based, patient-centric approach to healthcare delivery, enhancements to the current PCMM functionality are being requested that will allow a team to be formed and aligned around a patient, including providers across multiple VA sites and in non-VA settings to enable care coordination and communication. The software must also support automated data collection for management metrics and analysis related to access, workload, and panel management. This functionality would be ideally integrated into the future Clinical Practice Environment (CPE) versus being a separate and distinct module or application. VHA's model of team-based care is known as the PACT. The goal is to evolve or replace existing PCMM software application with functionality that identifies all team members and specialists (VA and non-VA) involved in the care of the patient, as well as their contact information and provide modalities to facilitate provider-to-provider communication.

When PCMM data is entered in a standardized manner, it can be used to analyze the system and PACT workload nationally by Veterans Integrated Service Network (VISN), and by a facility and its substations, as well as at the team level. PACTs manage the overall care provided to a majority of VA health care systems and their workload capacity is an important factor in determining the total number of patients that can be cared for in the system. In response to the growing number of Veterans wanting to use VA health care services, there is a need to quantify the PC capacity that is available so that demand and supply can be better aligned. PCMM allows users to set up and define a healthcare team, assign staff and health professions trainees to positions within the team, assign patients to the team, and assign patients to practitioners, including trainees. Data regarding PCMM team setup and assignments is used to calculate recommended panel size for PC teams and providers. The PCP and PC team information captured in PCMM is transmitted and stored at the Austin Corporate Franchise Datacenter (CFD), is available in the Corporate Data Warehouse (CDW), and is used for national reporting and performance measurement.

1.1. Purpose of this document

The purpose of this document is to describe in sufficient detail how the proposed system is to be constructed. The System Design Document translates the Requirement Specifications into a document

from which the developers can create the actual system. It identifies the top-level system architecture and identifies hardware, software, communication, and interface components.

1.2. Identification

Senior management within the Veterans Health Administration (VHA) and the Veterans Programs Portfolio has embraced two major principles – software methodology improvement and infrastructure modernization. The PCMM Rehost and Reengineering (“PCMMR”) project is an ideal example of incorporating both modern development processes, such as SCRUM-flavored Agile, and up-to-date tools and infrastructure, such as J2EE, Portal/Portlet functionality, Web 2.0, and a single enterprise database, into the redesign of the legacy client-server PCMM application.

1.3. Scope

Table 1 Scope Inclusions

Includes
High-level architectural components of PCMMR
External system interfacing to/from PCMMR
Basic PCMMR design ideology
CISS portal overview

Table 2 Scope Exclusion

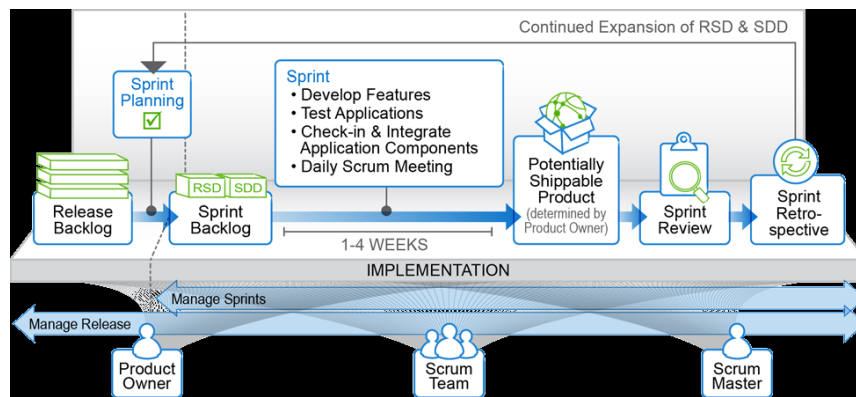
Excludes
Detailed data transmission protocols with external PCMMR systems
Detailed descriptions or analyses of external tools used in the development of PCMMR (see Background section 1.51.5 for examples)

1.4. Relationship to Other Plans

This document does not define functional requirements, the detailed Configuration Management plan, or the Software Quality Assurance Plan. Those will be described in other documents. The related RSD is **PCMMR_RSD.doc** (version 1.29 as of 2/19/2014).

1.5. Methodology, Tools, and Techniques

As mentioned in the introduction, the software development lifecycle (SDLC) used by the PCMMR team is the SCRUM version of Agile. This methodology breaks down the iterations of development into short sprints, which allow for fast adaptation to changing requirements and constant feedback and refinement to requirements from the business customer. Typical sprint activities are outlined in the following diagram:



To support its Agile SCRUM-based development methodology, the PCMMR team has decided to use VersionOne to support its planning, task management and scrum monitoring tasks. VersionOne is the leading project management tool designed specifically for agile software development. [REDACTED]

In day-to-day development activities, VersionOne tracks development test and analysis activities and monitors “burn-down”, the term used to describe the comparison of the expected progress within an iteration against the actual progress by the team. An example burn-down chart used by the team during the daily scrum meeting looks like:



Comparisons between sprints are also valuable to estimate the overall workload assumed by the team and ensure it’s not substantially higher or lower than what is reasonable. The team “velocity” is used to describe the average number of story points completed by the team from sprint to sprint. An example of the team velocity chart looks like:



1.6. Policies, Directives, and Procedures

This document, in conjunction with many others, has the additional purpose of satisfying the Veterans Administration Program Management Accountability System (PMAS) requirements. Only after these comprehensive requirements are met will the PCMMR VA customer allow the software production to be declared “active,” meet certain milestones, and eventually be deployed to one or more production environments.

In addition to PMAS procedures and deliverables, there are several more directives followed by PCMMR:

- Handbook 6500 describing VA information security standards
 - VA directive 6513 “Secure External Connections” doesn’t apply since PCMMR doesn’t access any systems external to the VA network and all PCMMR users must first be on the network to access the application (either directly or over VA VPN).
- KAAJEE Security for Authentication and Authorization - The VA mandates implementing J2EE application security through the use of the KAAJEE framework. This constraint will be taken into account when designing the technical solution.
- IAM compliance – PCMMR follows the technical and business standards published by the Identity and Access Management (IAM) group, to ensure that its user data remains protected and is consistent with the enterprise. Specifically PCMMR uses the following services:
 - Master Veteran Index (MVI) – serves as an authoritative source for persons’ identity traits. Provides initial VA identity correlation with external partners and across VistA sites, and maintains a record locator service for all client records known in VA.
 - Identity Integration – Ensures application integration and consumption of identity services is configured to ensure identity data remains current and accurate.
 - VAAFI – PCMMR access to MVI is secured via the VAAFI security tool

For more information, refer to [REDACTED]

- Java Coding Standards v2.0 - [REDACTED]
- Section 508 Checklist for Web-based Internet Information and Applications - http://www.ehealth.va.gov/508/terms/web_508_checklist.doc

1.7. Constraints

Business needs one, two, and eight of the PCMMR PWS must be satisfied to meet the initial prototype (POC) contractual requirements. Additional business needs will be satisfied and system functionality built during future development, such as the Initial Operating Capability at the one-year mark (IOC) and a future potential Option Year One (OY1).

The system must be a central large-scale enterprise deployment which consolidates many separate and disparate VistA data sources within the existing VA infrastructure. This consolidation effort will provide a more standardized, uniform interface to VA users as well as give them the capability of reporting on regional and national levels.

The system will be built using Java Enterprise Edition (JEE) components, portal/portlet code frameworks and UI tools to recreate and enhance the legacy PCMM application in a web-based format.

1.7.1.TRM

These tools are compatible with the VA Technical Reference Model (TRM) located at [TRMHomePage.asp](#) :

Application Server	WebLogic Portal Server 10.3.6	Approved
Database Server	Microsoft SQL Server 2008 R2 SP1	Approved
Database Driver	Microsoft SQL Server JDBC Driver	Approved
SQL Admin tool	Toad	Approved
Java Runtime	Java 1.7	Approved
Middleware	Spring Framework 4	Approved
ORM	Hibernate 4 / JPA	Approved
GUI	Javascript	Approved
HL7 Interface Engine	Mirth Connect	Approved

1.7.2.Release Architecture

The PCMMR production environment is compatible with the VA Release Architecture v1.21:

Operating System	Red Hat Enterprise Linux 5.8	<i>Current</i>
Platform	VMWare vSphere 5.2 virtual host	Current
System Availability: High Availability	vSphere Host Replication in both the PROD and DR sites; SQL Server database replication between DB hosts	Current
System Availability: Disaster Recovery	Identical standby disaster recovery environment located at Hines data center for the entire CISS portal and all portlet applications, including PCMMR	Current
System Availability: Scalability	PCMMR is designed as a clustered remote portlet, whose capacity can be extended by adding more instances of the application to the cluster	Current

Storage Technologies: Storage Design	OS uses RAID storage subsystem which meets the storage standards listed. Database is on a Storage Area Network with de-duplication	Current
Storage Technologies: Backup/Restore data	SAN-level backups	Current
Network: Local Area Network	Cisco hardware with remote management and a capacity exceeding PCMMR data requirements	Current
Network: Wide Area Network	Cisco hardware with remote management and a capacity exceeding PCMMR data requirements	Current
Client: Desktop/Laptop Standard Configuration	Standard Government-Furnished Equipment (GFE) builds (bundled with Internet Explorer) are compatible with PCMMR web application	Current
Client: Application Virtualization	N/A	PCMMR is accessed via web browser, not over a remote desktop or virtualized environment
Database Products	Microsoft SQL Server 2012	Current

1.8. Design Trade-offs

The existing legacy PCMM application is a client-side native application running within Microsoft Windows. By converting this functionality to a browser-based format, the user interface may not be as responsive across page refreshes as was, for example, clicking several tabs and buttons to navigate between teams or positions in the old system. However, the browser-based approach aims to provide a cleaner, less cluttered GUI for the end-user. Also, it is accessible on any platform, across many web browsers, and without needing to install any client-side software.

Data from PCMMR is made accessible to external systems for reference. Ideally, all systems would request data from the enterprise PCMMR application and be returned immediate results with up-to-date data. Unfortunately, some of these systems (e.g. CPRS) need to access this data at a fast enough rate that the enterprise installation would be overloaded and unable to meet the demand. The PCMMR team will accommodate these dependent systems by using cached copies of the data (local to each VistA site), and update those caches at a reasonable rate. The patient data synchronized to VistA is dependent on which patients actually exist in that VistA site, which is determined authoritatively by the MVI.

1.9. User Characteristics

The user community for both the legacy PCMM and web-based PCMMR systems consists of users who assume a variety of roles. These roles include, but are not limited to, national administrators, administrative associates, coordinators, configuration managers, facility and VISN coordinators, traveling veteran coordinators, and the associated backups for these roles.

It is expected PCMMR users will need specific training for the new system design and features.

1.9.1. User Problem Statement

As described in the introduction, the PCMMR software is an important component in measuring patient demand and PCP capacity to meet that demand, as well as reduce wait times. The legacy PCMM system is site-centric and limited in its capability to serve the patient. For example, if a patient spends six months each year in one location and six months in another (a “snow bird”), they may receive care at two different VA facilities. Each site currently has a separate installation of PCMM containing different team definitions, providers, etc. Coordination of the patient between these two facilities, as well as reporting of FTEE allocations and workload on the teams, is a manual process performed by PCMM users and is hampered by the inability to synchronize the different data sets. The legacy PCMM system doesn’t provide up-to-the-minute regional or national reporting of a patient’s assignment to teams, but instead relies on reports provided by downstream systems (e.g. the CDW), which may be delayed a few days or more. The idea of redesigning the application to become more “patient-centric” means that regardless of where the patient goes, a consistent set of data and medical history is available to PCMM users at all VA sites.

In addition to the mismatch between current functionality and the VA’s PACT requirements, other specific problems have been identified. The legacy PCMM data model is very complex to query, and allows for inconsistencies between teams and positions over time. To address these inconsistencies, PCMM users have resorted to “workarounds”, such as using free-text naming conventions for the teams and positions to track what really should be a consistent terminology lookup item in the database. Because different sites use different naming conventions, additional manual synchronization and resolution work has to be performed by PCMM users any time a patient needs to be tracked across sites.

1.9.2. User Objectives

Modernization of the PCMM software is needed to create a patient-centric model, which will give PCMMR users the ability to provide consistent care and reporting against the data sets managed by PCMMR.

2. Background

In order to fully support a team-based, patient-centric approach to health care delivery, enhancements to PCMM are being requested that will allow a team to be formed and aligned around a patient, including providers across multiple VA sites and in non-VA settings to enable care coordination and communication. The software must also support automated data collection for management metrics and analysis related to access, workload, and panel management. This functionality would be ideally integrated into the future Clinical Practice Environment (CPE) versus being a separate and distinct module or application. VHA’s model of team-based care is known as the PACT. The goal is to evolve or replace existing PCMM software with a PACT functionality that identifies all team members and specialists (VA and non-VA) involved in the care of the patient, as well as their contact information and provide modalities to facilitate provider-provider communication.

According to the PCMM Modification Workgroup, the relationship between the patient and his or her PCP can help to enhance patient care and the treatment process. Patients who regularly visit the same PCP

receive better health care and use fewer health care resources because of the primary provider's ability to treat patients more efficiently and effectively. This request is in support of the Major Initiative, "[Design a Veteran-Centric Healthcare Model](#)", which seeks to provide software that identifies and aligns the entire care team around the patient, including providers across multiple VA sites and in non-VA settings to enable care coordination and communication.

The re-hosted version of the Primary Care Management Module will be called the Patient-Centered Management Module. The upgrades to the PCMM system will allow users to efficiently edit team, group, panel, and position information, while also ensuring that patients are appropriately assigned based on required level of care.

Change control: The PCMMR contract allows the team to utilize its own development environment. The team utilizes an open source tool for their code change control management which enables the distributed software development teams to efficiently version and share source code. The open source tool contains features like atomic change commits, a single version for each tree revision (as opposed to per-file) and solid integration into both Windows and the Eclipse IDE.

Bug & requirement tracking: Issue management (i.e. bug tracking) for PCMMR will be managed by using a defect management tool which provides the complete scope of requirements and issue tracking as needed by the PCMMR HP development team and will also manage issues reported by users during the PCMMR User Acceptance Testing periods.

From the website: "Available in both a starter and enterprise edition, HP Quality Center software is a scalable, unified platform for managing and automating the delivery of secure, reliable, quality applications. HP Quality Center software enables you to implement a complete quality management infrastructure, establish consistent, repeatable processes and best practices for managing requirements, tests, and business components." [REDACTED]

2.1. Overview of the System

The purpose of PCMMR (both the existing legacy application and the redesign) is to manage the definition of teams of health care providers and the association of those teams to patients within the VA. PCMMR provides a user interface which allows authorized administrators to create teams, search for and add/remove providers to those teams, add patients to teams and provide a consistent snapshot of a patient's allocated set of team members over time, in alignment to the greater VA's desire for "Patient-Aligned Care Teams" (PACT). The purpose for the PCMM redesign is to migrate away from a large number of separate site-specific installations of PCMM and into a single national installation, which provides more consistent reporting of the patient across sites. The redesign also is a reengineering effort, migrating away from a "roll-and-scroll" console style application and into a web browser-based user interface.

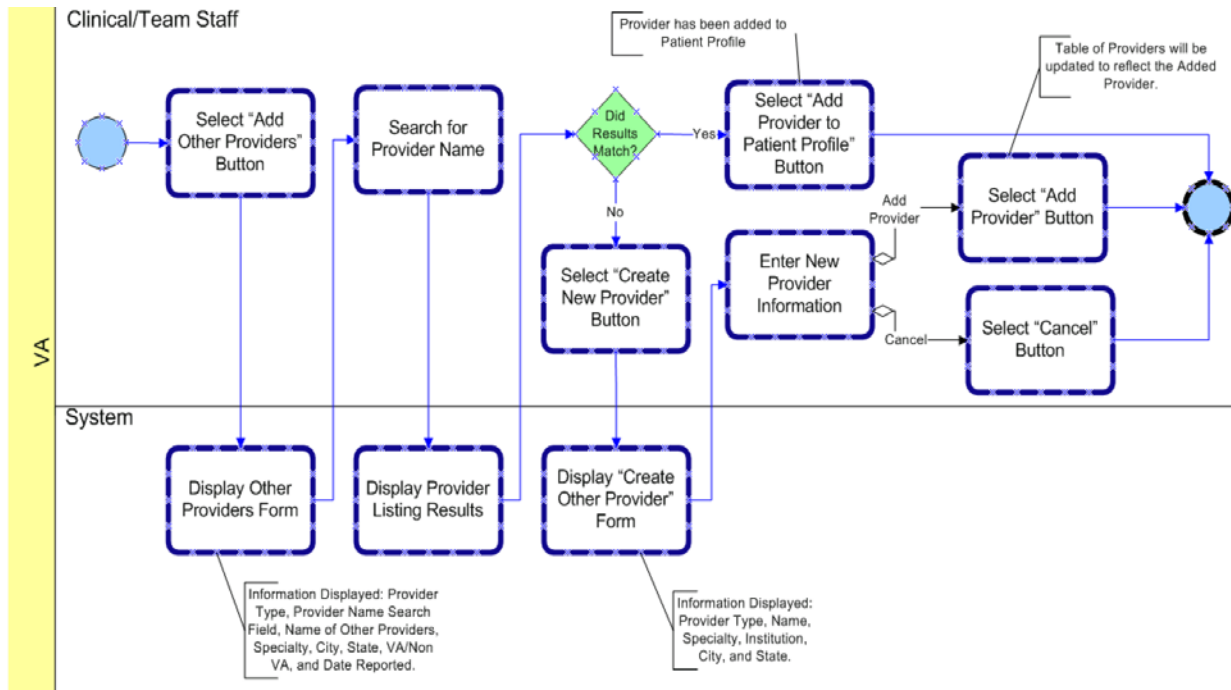
One or more users at each physical site (hospital, clinic, etc) in the VA hold roles within the PCMM application (such as PCMM Coordinator, Traveling Veteran Coordinator, VISN Coordinator, and National Administrator). Each role has associated access privileges which allow the person filling that role to perform functions within PCMM. The business process/flows for patient assignment in PCMMR is outlined in the following embedded document:



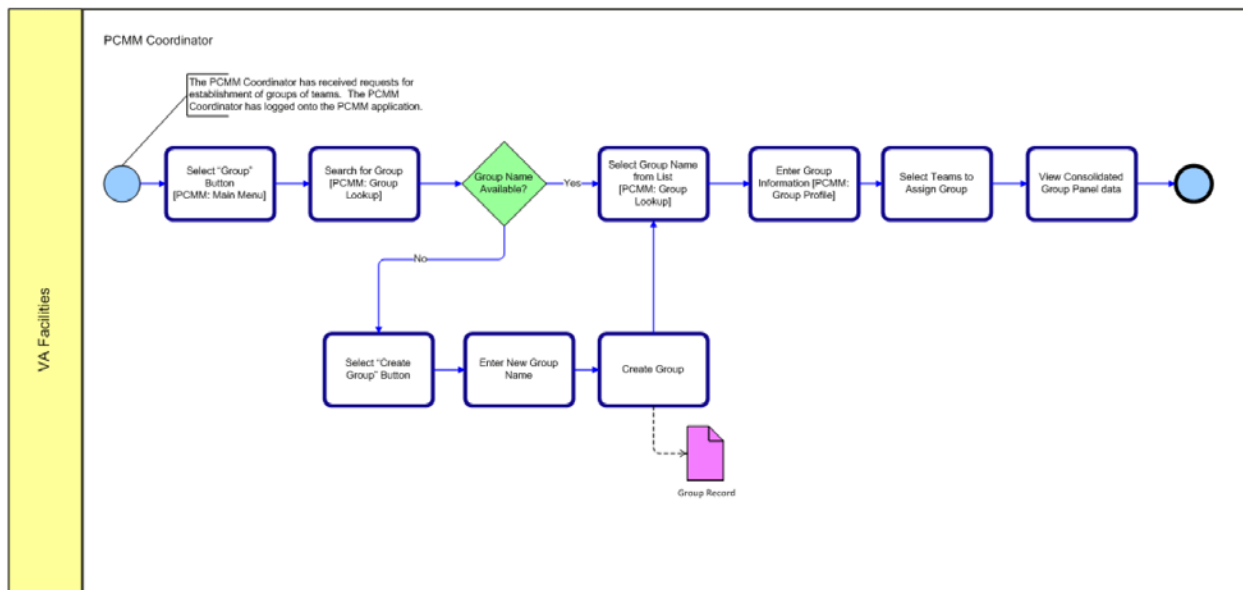
PatientNeedsAssign
ment.vsd

2.2. Overview of the Business Process

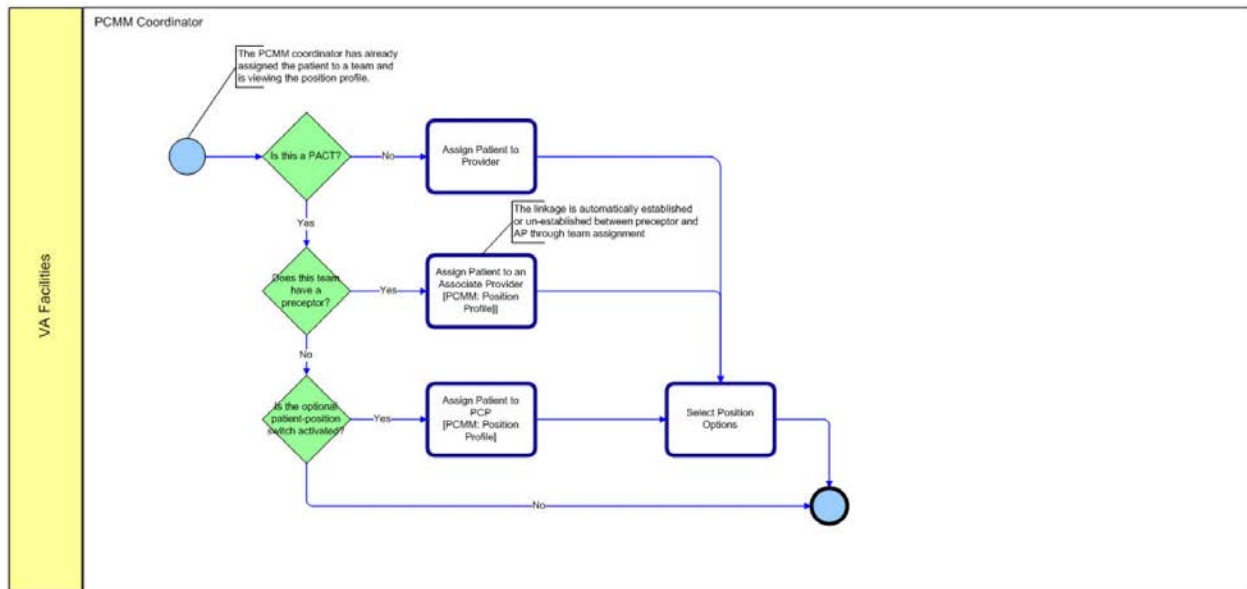
Below are several business process diagrams accompanied by a table describing their users and purposes. Cross reference the ID of each diagram with the ID in the table.



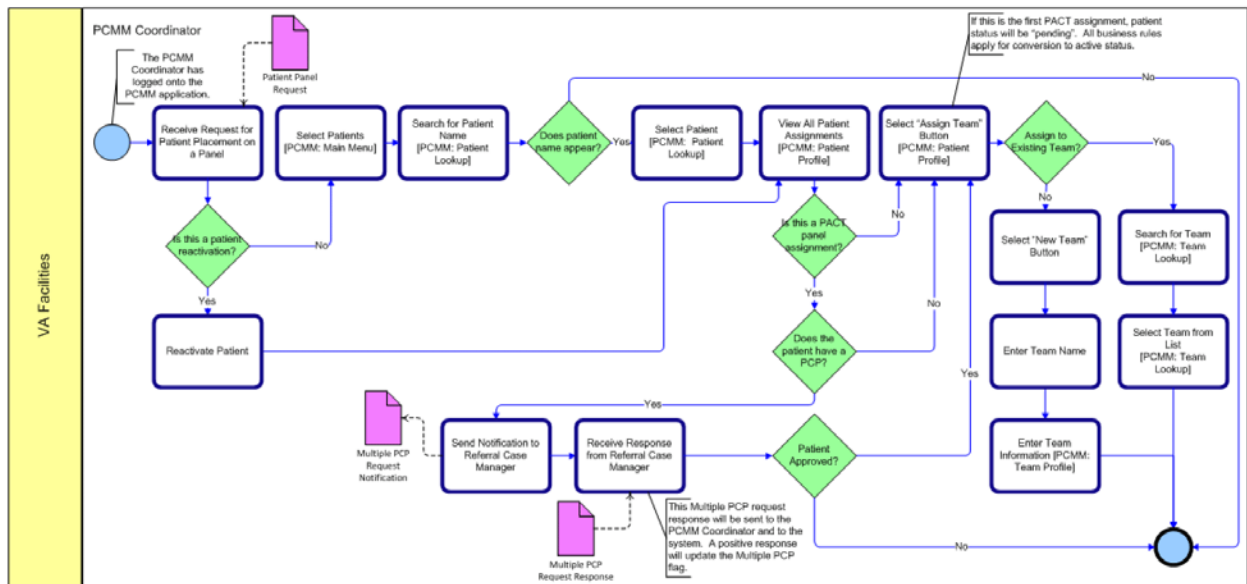
BPD1 – Add Other Providers



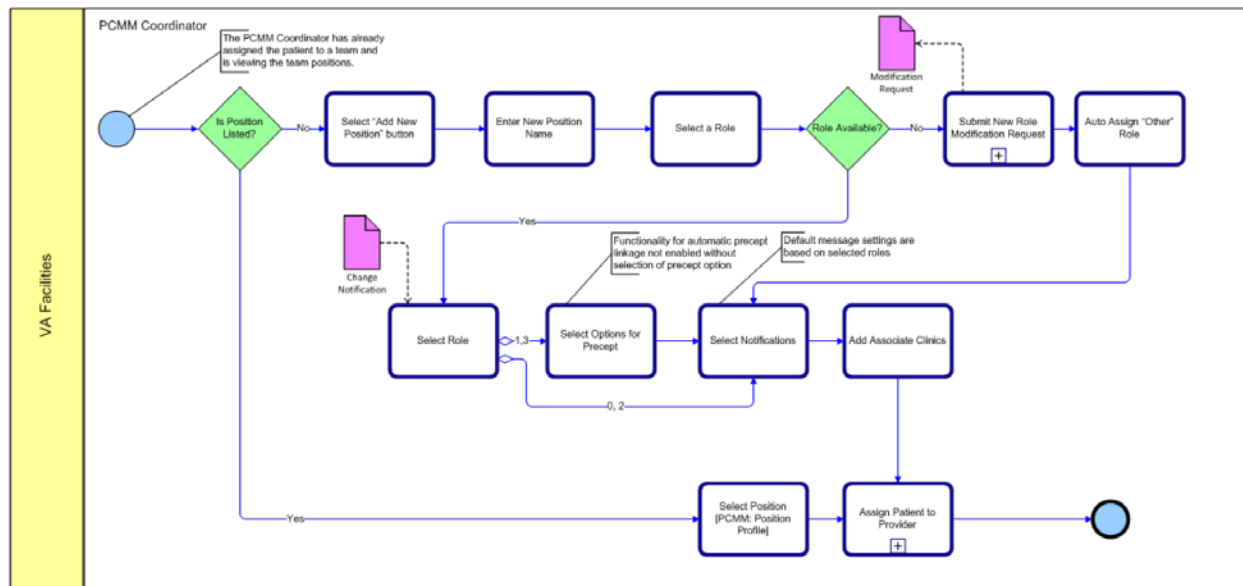
BPD2 – Assign Care Team to New Group



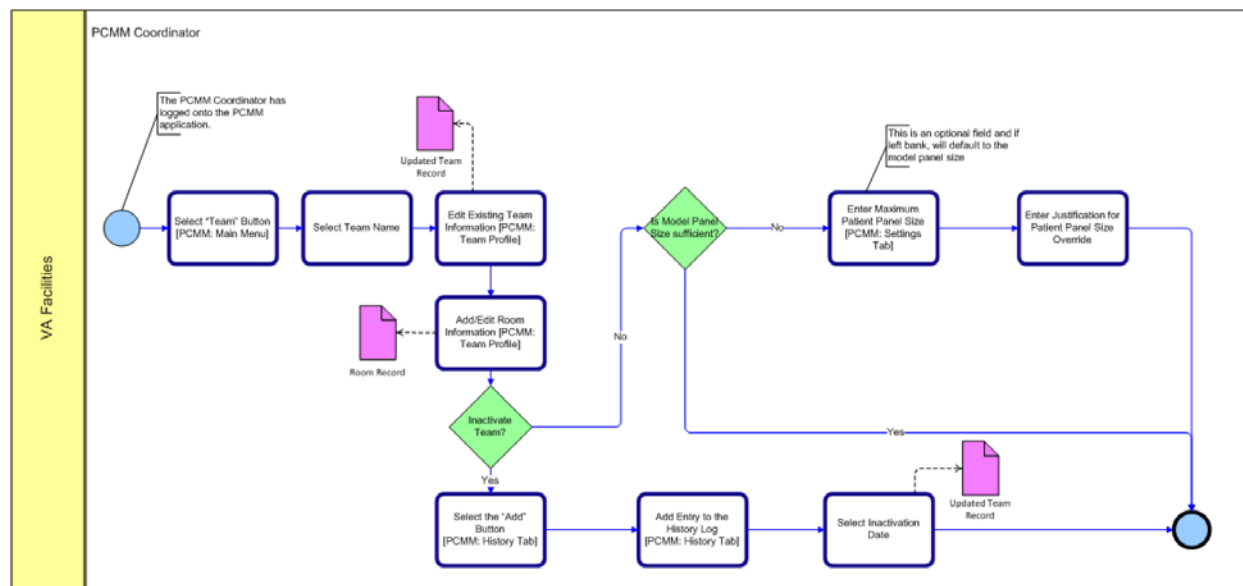
BPD3 – Assign Patient to Provider



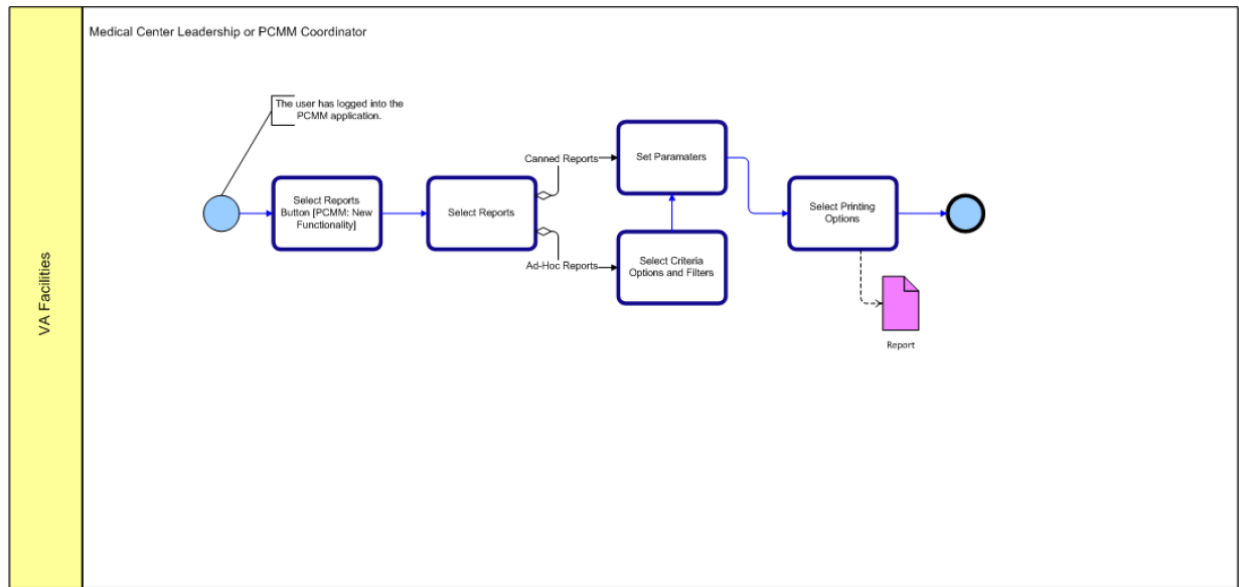
BPD4 – Assign Patient to Team



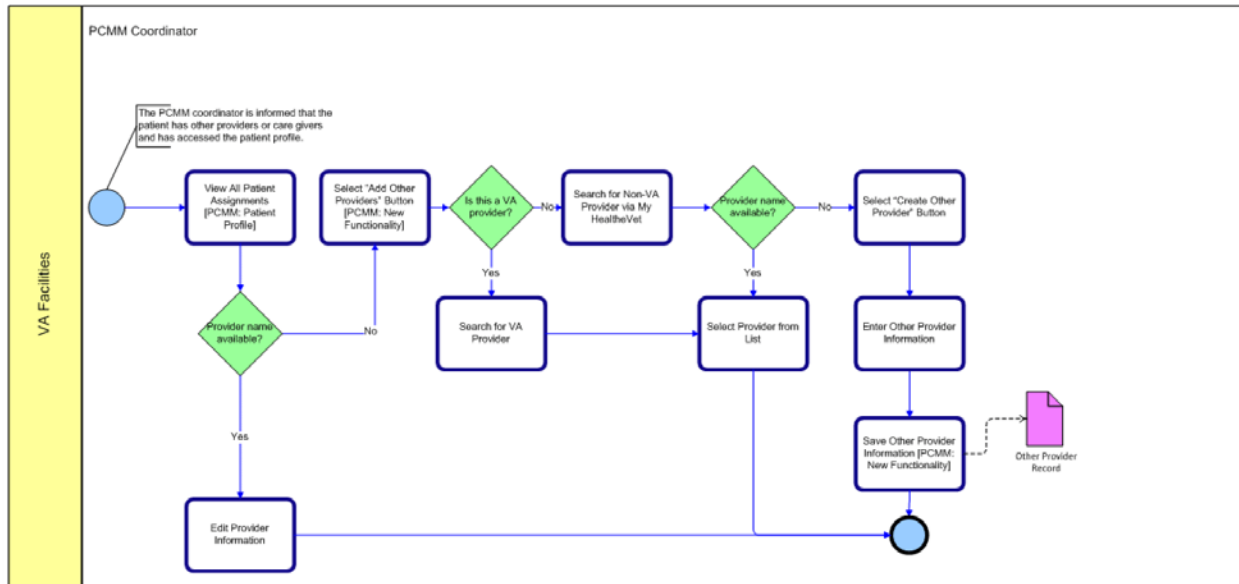
BPD5 – Assign Positions to Team



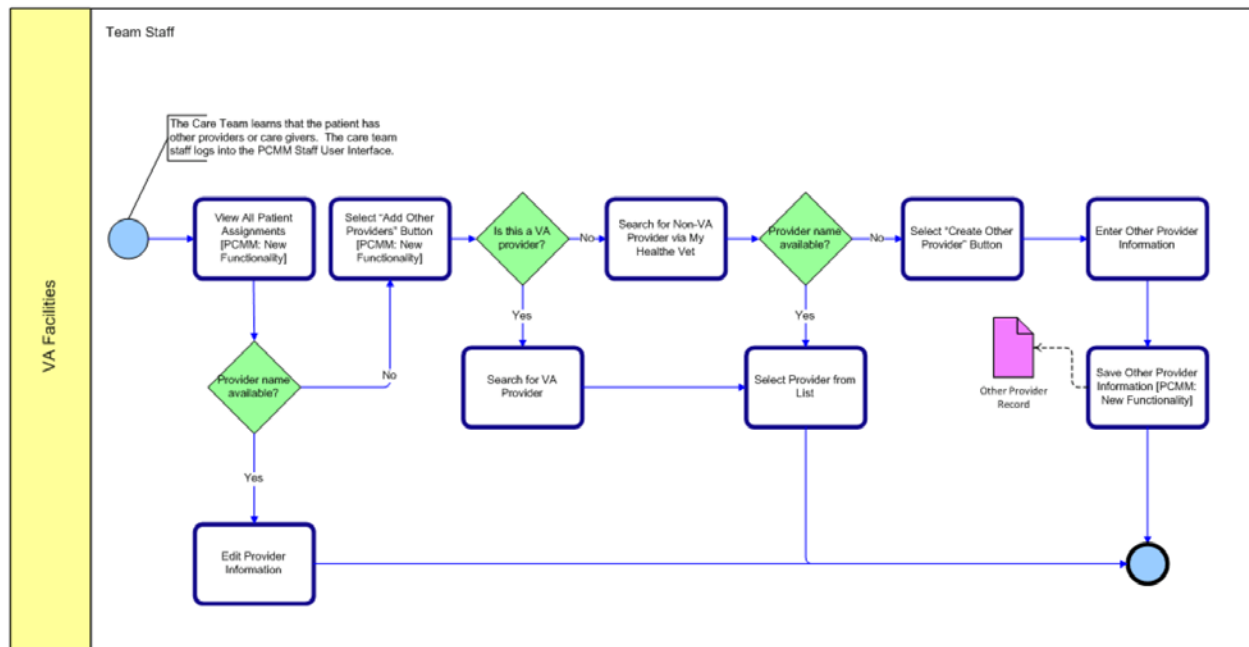
BPD6 – Edit Care Team Profile



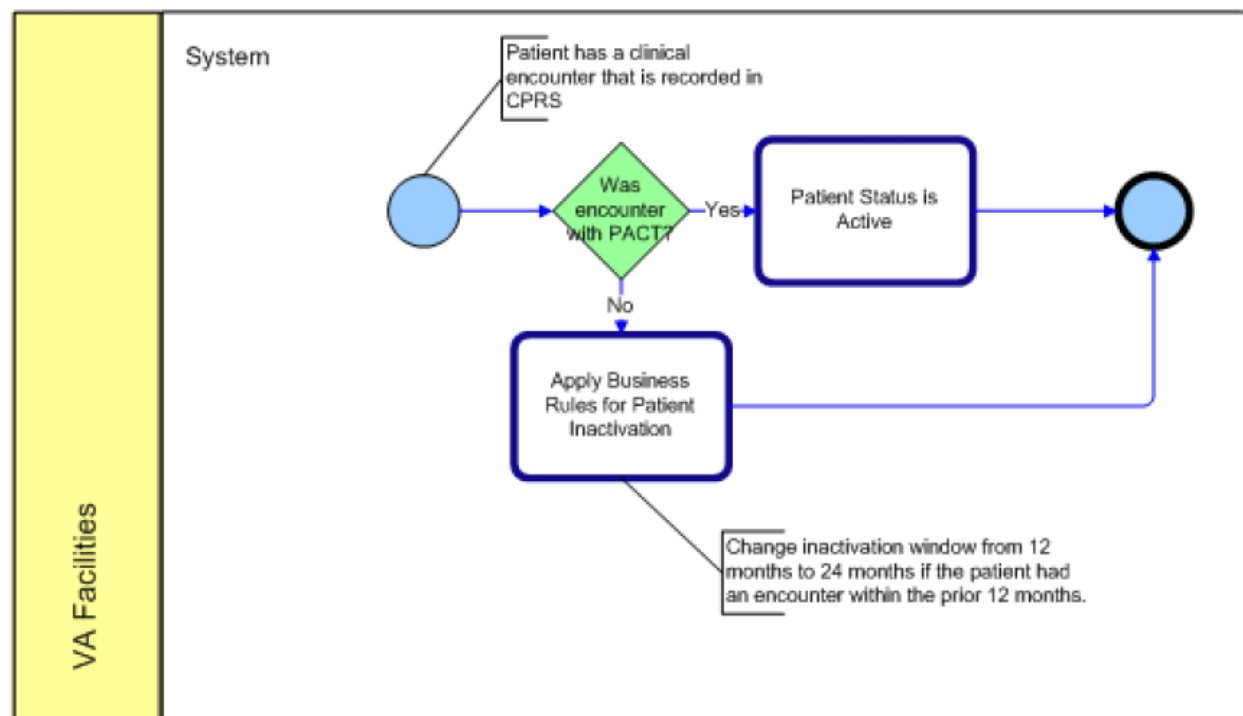
BPD7 – Generate PCMMR Report



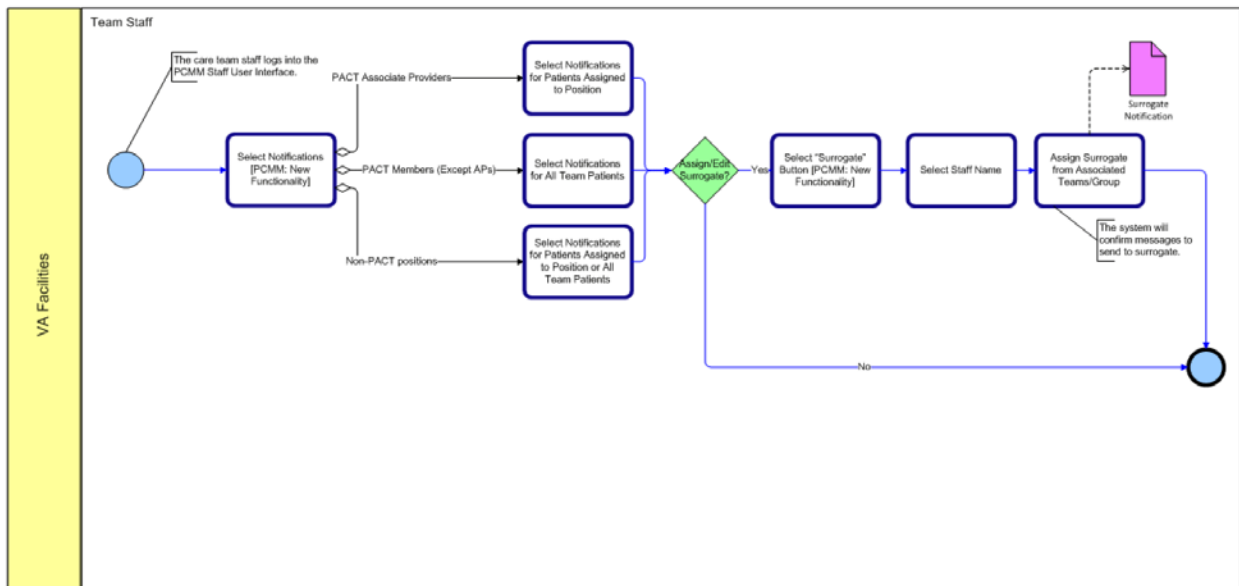
BPD8 – Identify Other Providers (PCMMR Coordinator point of view)



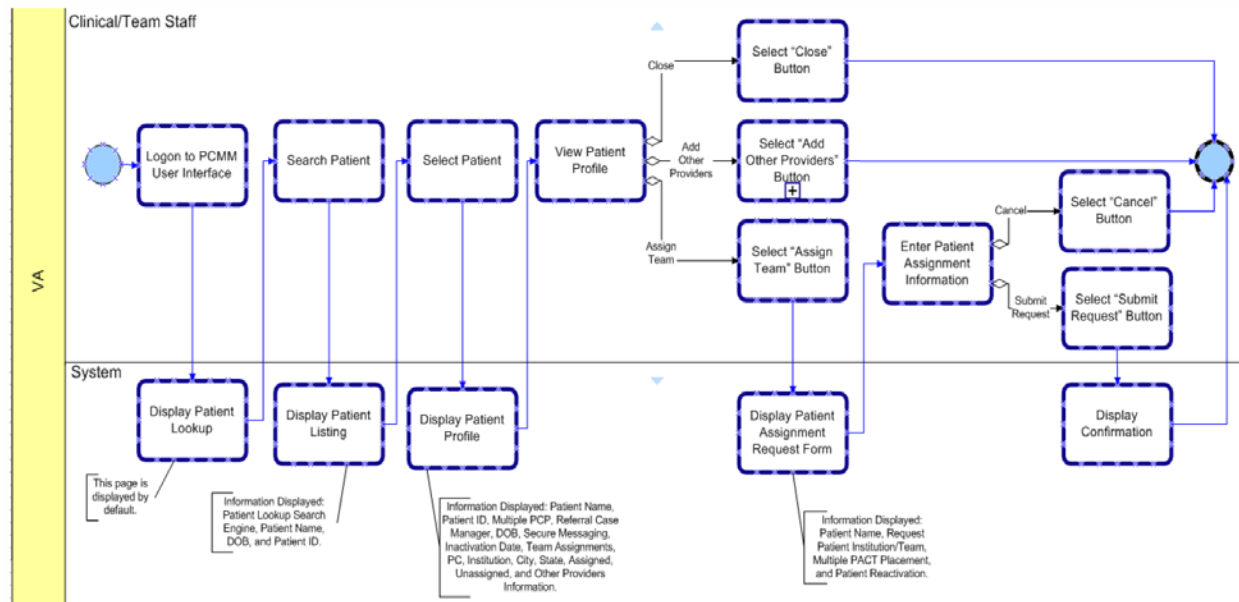
BPD9 – Identify Other Providers (Team Staff point of view)



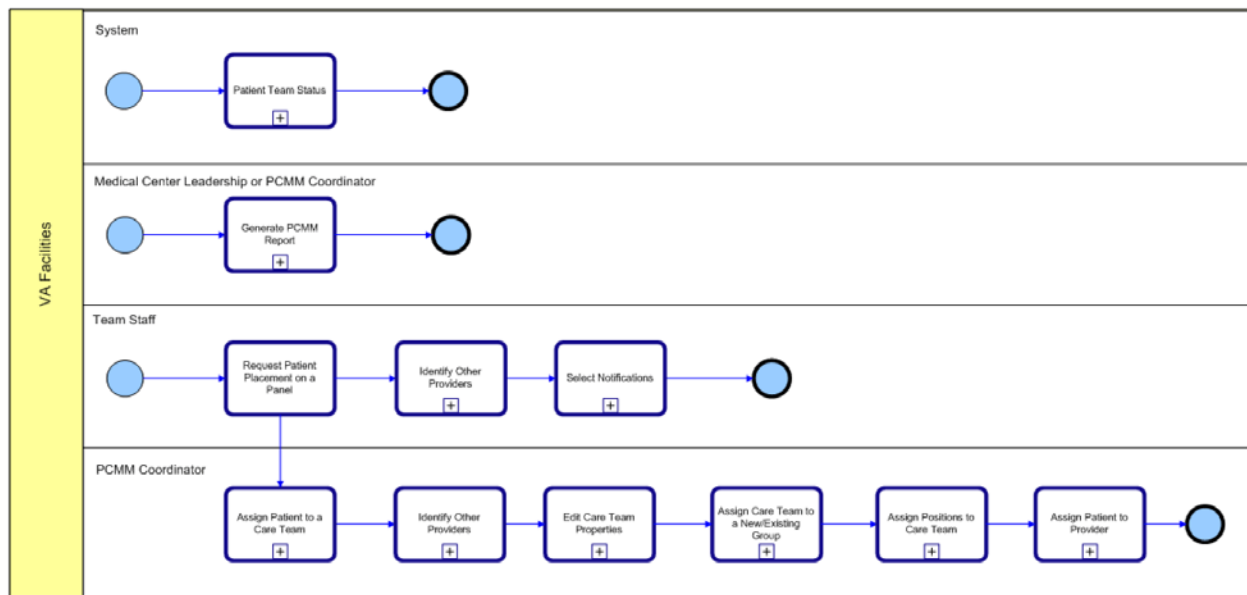
BPD10 – Patient Team Status



BPD11 – Select Notifications



BPD12 – User Interface



BPD13 – User Interface (by role)

Table 3 Business Processes

Business Process ID	Business Process Name	Type	Owner	Description
1	Add Other Providers	Modernized	All Users	General provider assignment to patient. This may or may not be implemented in PCMMR due to business requirements.
2	Assign Care Team to New Group	Modernized	All Users	In this model, Teams can be grouped together and this shows the process by which a team is assigned to a group. Groups may or may not be implemented in PCMMR due to additional ways to easily search and organize Teams.
3	Assign Patient to Provider	Modernized	All Users	This diagram shows how to assign a patient to a provider on a team. This generally was revamped in PCMMR such that a patient is directly assigned to a Team, and can optionally then be assigned to a PCP or associate on the team.
4	Assign Patient to Team	Modernized	PCMMR Coordinator	The PCMMR Coordinator at a specific station can assign patients to existing teams, or create a new team if one doesn't exist that's appropriate for the patient. Based on various logic as to whether the patient has one or more Primary Care Providers (PCPs), different options can be taken and parties notified (such as the Traveling Veteran Coordinator, previously referred to as the Referral Case Manager, in the event of a Multi-PCP request).
5	Assign Positions to Team	Modernized	All Users	Describes how to assign a new Position to a Team. Each position must be assigned a Role, but need not immediately have a Staff member

				assigned. Although not built for the prototype, Notifications will eventually be selected. Clinics will not be implemented per the business.
6	Edit Care Team Profile	Modernized	All Users	This flow describes the process of editing a Team. Room information will not be stored per the business.
7	Generate PCMMR Report	Modernized	All Users	This flow describes how users will create reports. Canned reports follow the process shown except that the “printing options” (which probably means the file type – PDF, XLS, etc) are integrated into the same Report Parameters page as other report inputs. Ad-Hoc reporting and scheduled reports are both future enhancements.
8	Identify Other Providers	Modernized	PCMMR Coordinator	This describes the method by which a PCMMR Coordinator can receive a request to add an external VA or non-VA provider to an individual patient that does not reside in the normal care team for the patient.
9	Identify Other Providers	Modernized	Team Staff	This flow is identical to the previous flow except that it represents the action of adding an external VA or non-VA provider to a patient from the Team Staff point of view.
10	Patient Team Status	Modernized	All Users	This flow describes how certain items in PCMMR will become inactivated automatically, unless an encounter is recorded in CPRS. During reactivation, if the encounter was with the PACT team defined and managed within PCMMR, the patient status is automatically set to “active,” otherwise a set of business rules is activated based on the encounter to determine whether to reactivate the patient.
11	Select Notifications	Modernized	All Users	This describes the process by which Notifications can be configured by the system. The Surrogate function allows for a different VA staff member to take over as recipient to some notifications.
12	User Interface	Modernized	All Users	This flow represents the typical user experience provided by PCMMR when editing a patient. The user logs in, searches for the patient and is taken to his or her profile. Then the user assigns the patient to a Team or to non-PACT providers.
13	User Interface	Modernized	Various Roles	From the point of view of various other roles, PCMMR can be used in several ways. The system may automatically update the Patient/Team status based on CPRS encounter data (see item 10 above). The PCMMR Coordinator generally will run reports. Team

				Staff members will assign patients to teams and external non-PACT team providers (see item 12 above). PCMMR Coordinators will manage Teams, Positions and Patients (although Groups will not be developed)
--	--	--	--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

2.3. Business Benefits

One of the business problems with the legacy PCMM application is the fact that PCMM coordinators at different sites enter data using separate local naming standards and business processes. For example, to add explicit links between patients and providers, some sites create entirely new teams and reassign sets of patients in bulk, whereas other sites rename or create new positions within the original team. It becomes challenging for the VA to provide sensical reporting across sites, or for sites to communicate intelligently with each other about patient allocations. The VA will benefit from PCMMR since it can save money previously wasted on users coordinating differences between sites and data analysts spending lots of time cleaning up data for accurate reporting.

Installation and maintenance of the software is another advantage. By having a single enterprise installation, the VA won't need to slowly coordinate/synchronize 130 separate application upgrades, or pay system administrators at each separate site responsible for the VistA patch. Also, since PCMMR is built using J2EE tools rather than MUMPS/Delphi, the VA gains the flexibility of selecting amongst a large set of development teams for future maintenance.

Finally, the data quality in PCMMR will be better than in the legacy application since it will integrate with the Master Veteran Index (MVI) data set, managed by the Identity and Access Management (IAM) team. This index provides an authoritative source of identity information about all VA patients. Currently, each VistA site contains a separate copy of patients. Problems can arise when PCMM coordinators search for patients who have recently changed their name or social security number, requiring extra time to resolve the differences and clean up data. By synchronizing its data with MVI, PCMMR allows the users to search against the latest updated set of patient identity information, and prevents the need for the VA to spend time and money resolving inconsistencies.

2.4. Assumptions and Constraints

2.4.1. Design Assumptions

- Since PCMMR will be deployed as a portlet partner application to the shared parent CISS portal, common data elements (e.g. users, profiles, SDS lookup tables) will be shared and cross-referenced within a single database and all PCMMR data will reside within that database as an independent schema.
- VA Network connectivity will be unimpeded between users' specific sites and the final deployment location of the CISS portal application. The PCMMR portlet application will be connected to the CISS portal behind the scenes, but users will not need network access to the specific machines that host the PCMMR remote portlet.
- Disabled users will continue to use screen readers and other assistance tools compatible with Section 508 to access all functionality of the final PCMMR system. This is consistent with the plan for PCMMR to be fully 508-compliant, as noted in section 1.6.
- The existing design of the legacy PCMM application which makes heavy use of tabs and Windows GUI interface elements will be replaced with a new screen-by-screen webpage interface. This new interface will provide all existing PCMM functionality, but in a more separated, "clean" approach.
- Military time will be the standard for UI input and display. All times will be entered and displayed in the user's local timezone, which is synchronized with the user's session on login to the application.
- The VA standard inactive period of 15 minutes applies to this system, and the user will be automatically logged out unless they choose to remain active.

2.4.2. Design Constraints

Although the existing CISS partner application, OHRS, was developed in Adobe Flex, support for this product has recently been discontinued by Adobe in favor of HTML5 and other dynamic front-end tools¹. Because of this shift (and due to the extra development effort and cost needed² to use Flex), the PCMMR development team chose to adopt the Spring Portlet MVC/JSP/JSTL tool stack for server-side presentation layer code and jQuery JavaScript framework for its lightweight client-side needs. The former set of tools was chosen for the following reasons:

- Tool maturity – Spring has an active developer community and has existed for years across the industry. It has become the framework of choice for many web application developers.
- Standards-based – JSP and JSTL are first-class members of the latest J2EE standard³ and are among the most common tools used for client-side markup generation.⁴
- Familiarity with development team – All developers on the PCMMR team are well-versed in these tools.

■ [REDACTED]

■ [REDACTED]

■ [REDACTED]

■ [REDACTED]

- CISS framework consistency – The CISS framework uses Spring, JSP, and JSTL, so continuing to use these tools keeps the codebase consistent.

jQuery became the JavaScript framework of choice due to the following reasons:

- Maturity – jQuery has existed since 2006. It is used by over 55% of the 10,000 most visited websites and is the most popular [JavaScript library](#) in use today.⁵ It has an active development community and forums for support.
- Cross-browser compatibility layer and CSS3 compliant – The fact that jQuery works across multiple web browsers is particularly valuable to the VA, since the VA allows for Internet Explorer versions 6-9.
- High number of overall features compared to other JavaScript frameworks.⁶
- Simple notation – Very little code is necessary to write powerful jQuery expressions.
- Modular– The jQuery framework allows for custom plugins to be used for enhanced functionality. If the VA needed to enhance or patch jQuery code later, it could do so under its own custom module.
- License flexibility – jQuery offers end users complete flexibility to modify, enhance, and redistribute any or all of its files as long as the copyright header is left intact, per the MIT License.⁷

PCMMR is a JSR-286 portlet which means it must abide by the portlet specification. Part of the specification is a multi-stage rendering process, during which the WebLogic application container manages things like form submissions, the transfer of events to and from other portlets, and requesting markup to be delivered from each portlet for final page rendering. Due to this complex rendering process, PCMMR needs to work around several small issues to accommodate its advanced requirements. These workarounds include support for AJAX calls by the front-end jQuery framework, connecting the HTTP session handling, establishing a shared CISS context with other portlets (which contains the logged-in user, selected timezone, and duty station) via Portlet event handling, and retrofitting several Servlet-based designs into the portlet space.

2.5. Overview of the Significant Requirements

2.5.1. Overview of Significant Functional Requirements

Table 4 Functional Requirements

ID	Specific Requirement / Synopsis	Requirement
2		Provide the ability to create and maintain a VHA enterprise level PCMMR system (as opposed to 128 individual disconnected PC

⁵ <http://trends.builtwith.com/javascript/JQuery>

⁶ http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

⁷ <http://jquery.org/license/>

		services).
3		Provide the ability to view VHA and non-VHA PCP, Team and Team member information in the patient's Electronic Medical Record (EMR), i.e., CPRS.
4		Provide the ability to manage specialty care teams (i.e. Mental Health, Cardiology, Pulmonary, Long Term Care, etc.) within PCMMR.
5		Group teams together: PCMMR teams will be reorganized so that there is one PCP per Team and if this is done there is a need for a way to associate multiple PC Teams with the larger collaborative team (e.g. Blue Team) which represents a grouping of PC Providers and teams who work together.
6		Flag patients in PCMMR that have completed In-Person Authentication and are to communicate via secure messaging and display as icon next to patient listing.
7		Update the Veterans Handbook: Retrieve PC provider location information (e.g. phone numbers and facility address) from PCMMR.
8		Provide a Centralized Data Base.

2.5.2.Functional Workload and Functional Performance Requirements

Table 5 Workload and Performance Requirements

ID	Requirement
7.2.1.1	Primary Care, Mental Health, OEF/OIF, CAC(clinical reminders), Spinal Cord Injury, Medicine, Surgical, and Emergency Department, Home Based Primary Care, etc. This does not include the Non VA facilities. All may need access to rehosted PCMMR. Prediction is double the 2760, add 50 states and 5 territories (Philippines Is., Guam, Samoa, Puerto Rico, Virgin Is.) non-va facilities access.
7.2.1.1 - Performance	It is anticipated that PCMMR functionality will be used to manage coordination of care between Primary Care, Specialty Services, and special veteran populations; therefore we expect users at least quadruple to 2244 users simultaneously. (source: Specialty, MH, OEF, Referral Case Mgrs) Despite the PCMMR software traffic going directly to a central database, response times will remain unchanged.
7.2.1.2 - Capacity	Individual transactions vary greatly depending upon what is involved with each one. Examples include set up team, assigning and unassigning a patient to a team and provider, activate/inactivate teams, assign/unassign (patient) to team, and assign/unassign staff. These activities can take seconds up to approximately two hours. Estimation that the system shall support a minimum of 1000 simultaneous users performing write and

	read transactions that each average 30 KB per, during the estimated peak usage hours of 8am – 11pm EST during weekdays
7.2.1.2 - Capacity	The system shall respond to user actions within three seconds or less 95% of the time under normal user loads of 1000 simultaneous user requests, and within five seconds or less 90% of the time under peak loads.
7.2.1.2 - Capacity	The numbers of transactions are expected to grow as the new enhancement capabilities move through implementation.

The PCMMR project conforms to existing IT Infrastructure Standards since it is simply an extension of the CISS /OHRs project which itself conforms to those standards.

The tables below list all current NSR entries on file with the Infrastructure team, the custom Integration Agreements (IA) and Integration Control Registrations (ICR).

Service	Category	Status
SDS support MS SQLServer	Terminology	NSR #20081117 Submitted and Reviewed
KAAJEE VPID/DUZ specification	Common Services	NSR #20080903 Submitted and Reviewed
CISS access to private KAAJEE api	Common Services	ICR #5255 created by KAAJEE team and Approved
XUS KAAJEE GET USER INFO	Common Services	ICR #5256 created by KAAJEE team and Approved
XUS KAAJEE LOGOUT	Common Services	ICR #5257 created by KAAJEE team and Approved
Access to Central PAID data	Identity Management	MOU signed by Roy Coles in 11/2008
Access to Volunteer data in VSS	Identity Management	ICR #5258 created by CISS team and Approved; Follow-on MOU submitted by CISS in 11/2008

Note: KAAJEE items (and possibly other authentication/authorization components like LDAP) will be eventually replaced with Identity Management Access Services / Single Sign On (SSO) at the CISS framework level.

2.5.3. Operational Requirements

Table 6 Operational Requirements

ID	Requirement
7.2.1.2	The application will need access to the Austin Information Technology Center (AIRC), and is used by the VHA VSSC to derive a number of patient care statistics, measures, and other information related to PC Management in support of Health Care Operations in VHA. This application will also interface with CPRS, VistA scheduling, and VHA CDW

2.5.4. Overview of the Technical Requirements

Table 7 Technical Requirements

User Story	Requirement
Active Panel Report	As an authorized PCMM2 User, I want to be able to generate the Active Panel Report so that I may review capacity and FTEE for each team.
Adhoc and Canned Reporting	As an authorized PCMM2 System, I want to be able to access any canned or adhoc reports through the SQL Server Reporting Services (SSRS) so that I may run pre-existing reports or create new reports.
Assign Patient to a Team and Position	As an authorized PCMM2 User, I want to be able to assign a Patient to a team and allow the user to choose which provider position is to be assigned if it is a Primary Care Team so that I may assign/track patient care.
Assign Room to Team	As an authorized PCMM2 User, I want to be able to assign a room to a team, allow the user to choose the team to be assigned and view the room assignment so that I may assign/track room FTE. Also, I want to be able to unassign a team from a room.
Assign Staff to a Position	As an authorized PCMM2 User, I want to be able to assign a staff member to a position so that I may assign patients to it.
Assign/Unassign Team to Group	As an authorized PCMM2 User, I want to be able to assign a team to a group, allow the user to choose the team to be assigned and view the group assignment so that I may assign/track team assignments to groups. Also, I want to be able to unassign a team from a group and display the history.
Associate Patient to Non-VA Provider	As an authorized PCMM2 user, I want to be able to associate a Non-VA provider's information with a patient's profile.
Batch Auto-Inactivation	As the PCMM automated system, I want to interrogate the encounters a patient has with his providers and have a notice sent out at 120 days before a patient is at risk to be unassigned due to inactivity within pre-defined periods of time or a Date of Death entry being recorded from any teams/positions to which he is currently assigned so that the teams have availability to treat active patients.
Batch Historical Assign Mass Patients	As an authorized PCMM2 User (PCMM Coordinator), I want to be able to select one or many patients from historical assignments and assign to a team or team position using a form that allows a batch process to run so that I may continue to work while it is running in the background.
Batch Staff Update	As the PCMM automated system, I want to have up-to-date information on the providers captured in the PCMMR staff profile so that the correct information will display in PCMMR as well as CPRS.
Batch Transfer	As an authorized PCMM2 User (PCMM Coordinator), I want to be able to select

Unassign Mass Patients	one or many patients to transfer or unassign to/from a team or team position using a form that allows a batch process to run so that I may continue to work while it is running in the background.
Batch Job Execution Results	As an authorized PCMM2 User, I want to be able to submit potential long running tasks to batch so I may continue to work while it is running in the background.
CPRS Primary Care Header	As a CPRS user, I want the Primary Care area of the CPRS header to indicate if a patient is assigned to a Primary Care Team at more than one facility (dual); if a patient's assignment to a Primary Care Team is pending; or if a patient's assignment to a Primary Care Team is at the local facility or a remote facility so that this information can be utilized by the clinicians serving the patient. I also want to see at a glance if the patient is an Inpatient and/or assigned to a Mental Health team.
CPRS Primary Care Window	As a CPRS user, I want to see a display of the patient's assigned Care Teams with contact information in the window accessed via the Primary Care (PC) area of the CPRS Patient Chart so that this information can be utilized by the clinicians serving the patient.
Completed Reports	As an authorized PCMM2 User, I want to be able to view a report once it has been generated.
Create a Team Position Profile	As an authorized PCMM2 User, I want to be able to create a position profile for a selected team so that I may assign staff to it.
Create a Team Profile	As an authorized PCMM2 User, I want to be able to create a team so that I may assign rooms/roles/staff to it.
Create a Non-VA Provider	As an authorized PCMM2 user, I want to be able to manually enter and update/view a Non-VA provider's information into the PCMM system.
Create Notifications Distribution Section	As an authorized PCMM2 User (PCMM Coordinator), I want to be able to create notifications distribution information so that I can view the origination and settings for the notifications needed.
Create Room Profile	As an authorized PCMM2 user, I want to be able to create room profile information and add to the List All Rooms screen so that I can view the newly created rooms. Also, add a new sub-menu option for Create a Room Profile to allow the user this functionality from the main menu "Room."
Create Update Group Profile	As an authorized PCMM2 user, I want to be able to access a new menu option for Group that allows users to create a new Group and update a Group so that the group information can be added to PCMM system.
Home Page and Main Menu	As a PCMM Coordinator, I want to be able to login to PCMM using the CISS portal so I can read announcements and perform my tasks. Also, once in PCMM, I want to be able to access the links provided from the Main Menu 'Reference' tab so I can view policies/procedures and other publications.
Legacy PCMM to PCMMR Data Exchange	As an authorized PCMM2 System, I want to be able to interrogate the results of the Date of Death triggers initiated from VistA and write the results back to PCMM/R.
Login to PCMM	As an authorized PCMM2 User, I want to be able to be able to logon to PCMM so that I may utilize the application.
Login to CISS	As an authorized PCMM2 User, I want to be able to be able to logon to CISS so that I can launch PCMM.
Logout of PCMM/CISS	As an authorized PCMM2 User, I want to be able to be able to logout of PCMM/CISS so that I may exit the application.
Manage Alerts	As an authorized PCMM2 User, I want to be able to view a list of alerts for my station and my role so I can act on them accordingly
Manage Groups	As an authorized PCMM2 User, I want to be able to view a list of existing groups

	so that I may choose one to view detailed information.
Manage New User	As an authorized user I want to be able to create/update/inactivate/unlock a system user so that they can access PCMM/R and perform their authorized functions.
Manage Notifications Distribution	As an authorized PCMM2 User (PCMM Coordinator) , I want to be able to set up the distribution of notifications to the appropriate Team members and positions so that the staff assigned to the Teams and Positions will receive the Notifications for patients that are assigned to teams/positions independent of where the notification for the patient is triggered. (Dr Stark example –patient on PACT in NY goes to FL and gets admitted. Team/positions in NY will get notification of the admission that took place in FL.
Manage Reference Data for Selection Lists	As a PCMM National Coordinator I want to be able to modify (add, edit, delete/inactivate) items on the dropdown lists within PCMM, so that the selection list of valid values will not require modification by OI&T/developer involvement and will available for selection immediately after being made.
Manage Rooms	As an authorized PCMM2 User, I want to be able to view a list of existing rooms so that I may choose one to view, edit or delete information.
Manage Team Positions	As an authorized PCMM2 User, I want to be able to view a list of existing teams so that I may choose an existing position to view or update information or to create a new position.
Manage Teams	As an authorized PCMM2 User, I want to be able to view a list of existing teams so that I may choose one to view, edit or delete information for.
Modeled Team Capacity Calculations	As an authorized PCMM2 User, I want to be able to have the system calculate the Modeled Team Capacity so that each team may utilize it as their recommended panel size.
Multiple PCP Assignment Action	As an authorized PCMM2 User, I want to be able to designate the patient is approved/denied for Multiple Primary Care Provider (MPCP) assignment(s) and/or edit the form so that I may confirm the patient is eligible for primary care at other stations.
Multiple PCP Assignment Capture and Display	As an authorized PCMM2 User, I want to be able to designate the patient is eligible/approved for multiple Primary Care Provider (PCP) assignments so that I may confirm he is assigned to the correct Primary Care team(s) at the correct station(s).
Patient Merge from MVI	As an authorized PCMM2 System, I want to be able to receive and process update messages for each patient that PCMMR has registered for interest in that are received from MVI so PCMMR can keep current with MVI and other applications. This includes notification when the ICN is first assigned to a patient, an ICN is updated for a patient for a station/dfn, or when a VistA record merge occurs and PCMMR has registered interest that is not the primary source ID. This is known to MVI as Resolve Duplicate – Merge and these are sent to PCMMR on an A24 transaction. NOTE: There may be 2 different patient records with the same ICN for a period of time while these txns are running.
Patient Move from MVI	As an authorized PCMM2 System, I want to be able to receive and process update messages that are received from MVI for each patient that PCMMR has registered for interest in so PCMMR can keep current with MVI and other applications. This includes notification when one patient is moved from one ICN to another. MVI refers to this as ICN mismatch.
Patient Updates from MVI	As an authorized PCMM2 System, I want to be able to receive and process update messages for each patient that PCMMR has registered for interest in that are received from MVI so PCMMR can keep current with MVI and other applications.
PCMMR to Legacy PCMM Data	As an authorized PCMM2 System, I want to be able to send the teams and assignments entered into PCMM/R to PCMM so they will be available for VistA

Exchange	applications including CPRS.
Query Patient and Register Interest in MVI	As an authorized PCMM2 System, I want to be able to search MVI for a patient in the background so it can validate the patient is known to MVI, register PCMMR to receive updates, receive the list of treating facilities for the patient and receive the most current patient demographic information for display in PCMMR.
Report List	As an authorized PCMM2 User, I want to be able to view a list of standard reports from the Reports menu.
Report Parameters	As an authorized PCMM2 User, I want to be able to enter/select parameters for the selected report from the “Report List” menu.
Reporting – Sensitive Patient Access Log	As an authorized PCMM2 System, I want to be able to access the Sensitive Patient Access Log through the SQL Server Reporting Services (SSRS) so that I may view who has been accessing sensitive patients.
Restrict Access for Employee Viewing Own Data	As PCMM system, I want to be able to restrict an employee from viewing his own records so that I may fulfill VA requirements.
Search for Group	As an authorized PCMM2 User, I want to be able to access a menu option for Group for search to allow the user to perform a search for existing Groups within the PCMM system so that I can view the group’s information.
Search for Patient	As an authorized PCMM2 User, I want to be able to search for a patient so that I may update his profile information or assign him to a position/team.
Search for Room	As an authorized PCMM2 User, I want to be able to access a new menu option for Room search and allow the user to perform a search for a Room within the PCMM system.
Search for Non-VA Provider	As an authorized PCMM2 User, I want to be able to access a new menu option for Non-VA search and allow the user to perform a search for a Non-VA provider within the PCMM system.
Search or Maintain Model Team Configuration	As an authorized PCMM2 User, I want to be search for an existing Model Team Configuration or be able to create, view or maintain a Model Team Configuration so that it can be used to create default Team Positions when creating new Teams, allow teams to reconcile against the model, and to validate Teams adherence to the Model.
Search Staff by Name	As an authorized PCMM2 User, I want to be able to search for staff by name so that I may assign staff to a position.
Search Team by Name	As an authorized PCMM2 User, I want to be able to search for a team so that I may work with its profile information or assign him to a patient.
Unassign Patient from a Team	As an authorized PCMM2 User, I want to be able to completely unassign a Patient from a team he is currently assigned to so that I may re-assign him to another team or indicate care/team assignment is no longer needed.
Unassociate Patient from Non-VA Provider	As an authorized PCMM2 user, I want to be able to remove an association of a Non-VA provider from a patient’s profile.
Update Facility List from MVI	As an authorized PCMM2 System, I want to be able to receive and process update treating facility list messages for each patient that PCMMR has registered for interest in from MVI so PCMMR can keep current with MVI on all treating facilities that patient has a presence at. The treating facility list is a list of systems that know a specific Integration Control Number (ICN). The list can contain systems that are not VAMC like FHIE or HDR. PCMMR will only update the Treating Facilities.
Update a Team Position Profile	As an authorized PCMM2 User, I want to be able to update a Position’s profile so that I can keep it current.

Update a Team Profile	As an authorized PCMM2 User, I want to be able to update a team's profile so that I can keep it current.
Update Primary Care Intensity Score	As an authorized PCMM2 User, I want to be able to update the primary care intensity score for a station so that it can be used to adjust the modeled team capacity calculations at a station and team level.
View a Patient Profile	As an authorized PCMM2 User, I want to be able to view the profile for a patient so that I may confirm he is the patient I wish to work with and I may update/view his multi-PCP indicator.
View Aggregate Model Capacity	As an authorized PCMM2 User, I want to be view the aggregate Modeled Team Capacity (ie panel size) for each team care type at a station level so that I can analyze and validate the station and its team's adherence to the recommended model panel size.
View Panel Placement Request	As an authorized PCMM2 User, I want to be able to create a Panel Placement Request so that I may capture and display my request to the receiving team/station.
View Patients Assigned to a Position	As an authorized PCMM2 User, I want to be able to view the patients assigned to a position within a team so that I can see the patients assigned to each team member.
View Patients Assigned to a Team	As an authorized PCMM2 User, I want to be able to view the patients assigned to a team so that I can see the patients latest assignment details.
View Patient Assignment History	As an authorized PCMM2 User, I want to be able to view which teams/positions/staff members each patient was assigned to at any point in time so that I may validate which team and staff member was responsible for patient care during that period of time.
View Staff Profile	As an authorized PCMM2 User (PCMM Coordinator), I want to be able to search for VA staff member and display that staff member's information on the profile screen.
View Team in CPRS Patient Header	As an authorized PCMM2 User, I want to be able to view the patient's Primary Care team, provider, associate provider on the first line and the Mental Health Coordinator on the third line of the CPRS Header so that this information can be utilized by the clinicians serving the patient.
View Team in CPRS Primary Care Screen	As an authorized PCMM2 User, I want to be able to view the patient's Primary Care team, provider, associate provider and Mental Health Coordinator and contact info so that this information can be utilized by the clinicians serving the patient.

PCMMR is fully compliant with the One-VA Technical Reference Model (TRM), using only those tools which are approved without constraints, or tools having constraints but for which it meets those constraints (such as Maven). See section 1.7.1 for a listing of specific tools in the TRM.

PCMMR is also fully compliant with the VA Enterprise Architecture since it is only a sub-module participating in the larger CISS framework, which is itself compliant with the VA Enterprise Architecture. See section 1.7.2 for a listing of release architecture components.

2.5.5. Overview of the Security or Privacy Requirements

2.5.6. System Criticality and High Availability Requirements

Table 5 Availability Requirements

ID	Requirement
7.2.1.3 - Availability	Should be available 24/7/365 (100%) To ensure the most efficient and effective operational status this system needs to be available at all times and should be maintained with scheduled back up regularity and maintenance.

The PCMMR application will be deployed in a staged approach across the various nodes of the production cluster, such that existing nodes can service current user requests while other nodes are being upgraded. If database changes also are required, existing data will be backed up in its entirety before any upgrades are made.

PCMMR will have a detached, co-located disaster recovery (DR) environment alongside other similar DR environments for CISS and the other partner applications of CISS. In the event of a catastrophic outage at the primary production hosting facility, the secondary DR instance can stand in as an interim production system. A data transition / replication may need to occur prior to the DR environment coming online, to ensure the latest user changes were captured.

Additional features to ensure uptime are:

- At the OS level with the Virtual server VMware Hosts ability to move VM's from HOST to HOST with no interruptions.
- Duplication of VM Hosts between the Datacenters.
- Database Replication between the Datacenters in real time.
- Database Clustering at the Microsoft Operating System and Microsoft SQLServer Level.
- Weblogic Clustering.
- Load balancers that monitor the availability of the actual Weblogic JVM's.

2.6. Legacy System Retirement

Table 9 Proposed Legacy Retirements

Legacy System or Legacy System Component	Retired or Workload Reduced	If Workload Reduced – How Much
PCMM	Retired	N/A
VSSC / CDW	Workload Reduced	Some reporting provided by these tools may be now re-implemented as canned or ad-hoc reports within the new PCMMR

3. Conceptual Design

3.1. Conceptual Application Design

3.1.1. Application Context

Figure 1 PCMMR Application Context Diagram

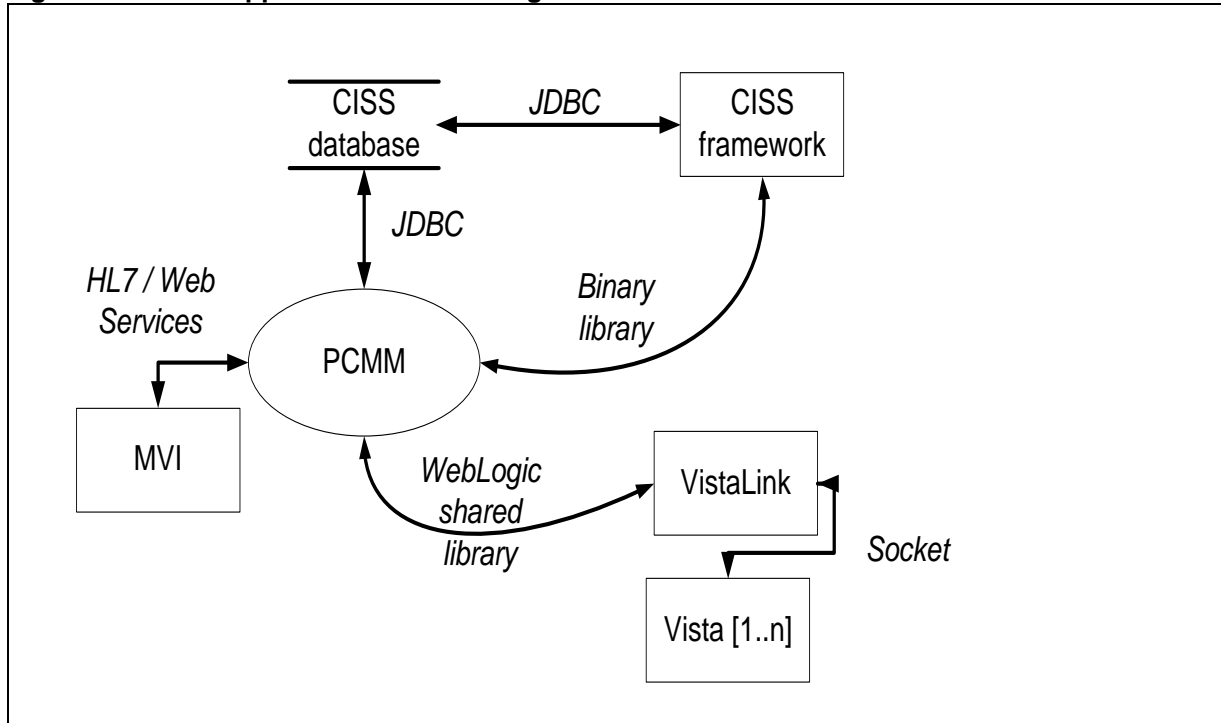


Table 10 Application Context Description

Objects			
Name	Description	Interface Name	Interface System
CISS Framework	Portal container which provides security layer (users, roles, permissions) and a portal login page which authenticates users against VA Active Directory before forwarding their session on to the PCMMR application	Direct code access thru shared library and typical servlet environment (no CCOW / SSO / WebServices)	CISS database, VistaLink
MVI	Authoritative source of veteran identity information	Web Services to query and retrieve patient data; HL7 message stream to capture identify management changes	Various systems throughout the VA also exchanging identity data
VistaLink	Provides a Java-compatible connection to a set of remote VistA systems	Deployed as a shared library (for access by applications) and also as a WebLogic resource adapter and console web application for monitoring	All remote VistA instances

3.1.2.High Level Application Design

Figure 2 Sample High Level Application Design

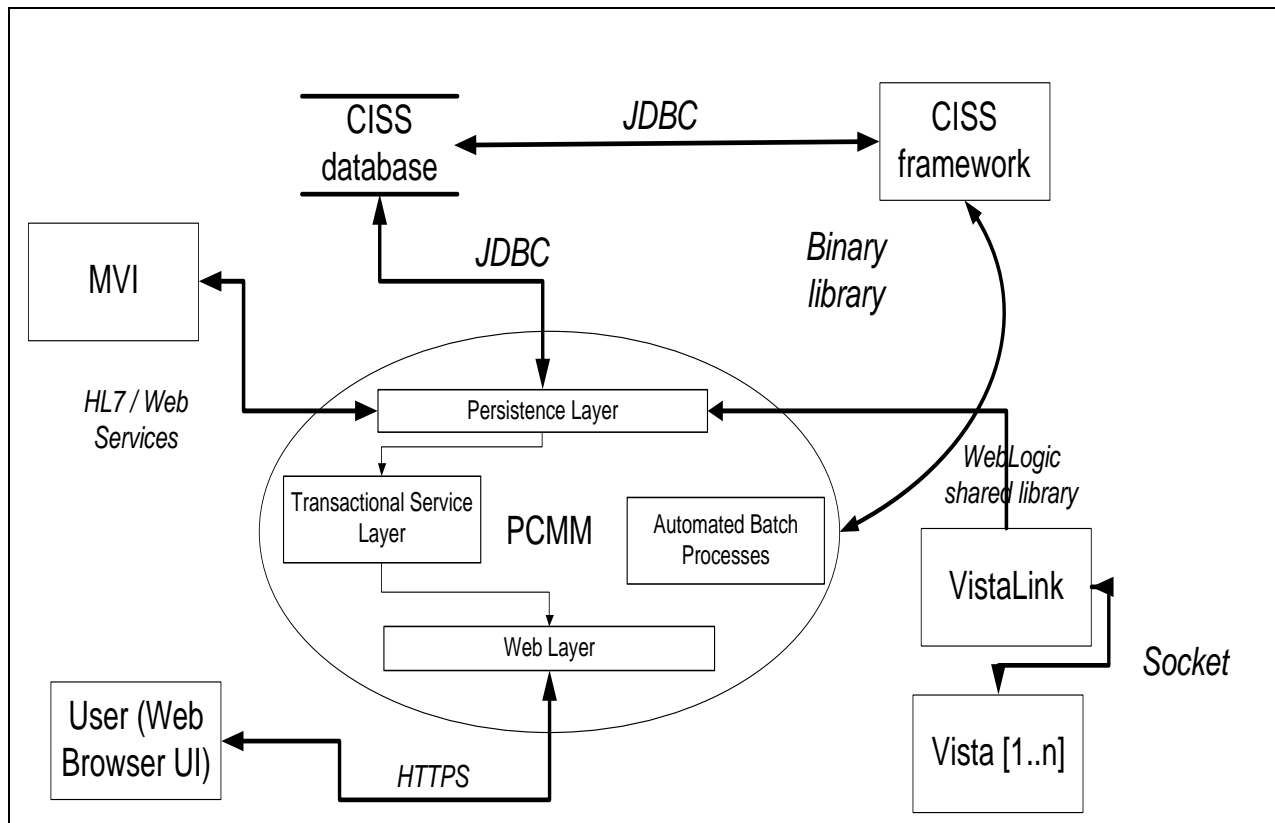


Table 11 Objects in the High-Level Application Design

Name	ID	Description	Service or Legacy Code	External Interface Name	External Interface ID	Internal Interface Name	SDP Sections 1&2
User (Web Browser UI)	1	Represents end users connecting to the enterprise PCMMR application via a web browser	New	Web layer	2	HTTP / HTTPS socket connections	Being Developed
Web layer	2	Contains the presentation tools and markup necessary to deliver the user interface to the users, and collect data from users when editing PCMMR objects	New	User (Web Browser UI)	1	Controller Java classes via Spring MVC, grouped in the gov.va.med.pcomm.web package	Being Developed
Transactional Service Layer	3	A set of objects performing all business logic and validations in the application. These items automatically create database transactions around their units of work, and operate in an atomic fashion.	New	None	N/A	*Service / *ServiceImpl Java classes inside the gov.va.med.pcomm.service package	Being Developed
Automated Batch Processes	4	A set of independent processes that perform various maintenance tasks for operational support in PCMMR	New	Headless	N/A	@Scheduled proxy class provided by Spring to activate classes via cron syntax or fixed delays	Being Developed
Persistence Layer	5	A set of objects whose responsibility is to abstract the data transmittal to/from various external data sources, such as VistA, LDAP, SQL database	New	None	N/A	*DAO / *DAOImpl Java classes inside the gov.va.med.pcomm.persistence package	Being Developed

3.1.3. Application Locations

Table 12 Application Locations

Application Component	Description	Location at Which Component is Run	Type
PCMMR Attended server	WebLogic domain instance which handles PCMMR user-facing (web) requests	Falling Waters, WV data center, on the same virtual machine as the CISS domain instance	EAR application file containing all layers of the application
PCMMR Unattended server(s)	WebLogic domain instance which handles batch processing and report execution	Falling Waters, WV data center, on the same virtual machine as the CISS domain instance	EAR application file containing the batch process and reporting components of PCMMR
Disaster Recovery PCMMR Attended Server	WebLogic domain instance which handles PCMMR user-facing (web) requests	Hines, OR data center, on the same virtual machine as the DR CISS domain instance	EAR application file containing all layers of the application
Disaster Recovery PCMMR Unattended server(s)	WebLogic domain instance which handles batch processing and report execution	Hines, OR data center, on the same virtual machine as the DR CISS domain instance	EAR application file containing the batch process and reporting components of PCMMR
VistA PCMMR module	RPCs and MUMPS code inside VistA which serves as a broker between the enterprise PCMMR instance and downstream systems	130 VistA sites across the US	Interface Code

3.1.4. Application Users

Table 13 Application Users

Application Component	Location	User
PCMMR Web Interface	130 VistA sites across the US	PCMMR coordinators, traveling veteran coordinators and all other application users
PCMMR Administration tools	Dependent on the user's location; accessible from anywhere on the VA network	PCMMR National Coordinator

3.2. Conceptual Data Design

3.2.1. Project Conceptual Data Model

The current data model is depicted in the PCMM_DataModel.pdf document, in the source control repository for PCMMR under the docs/A&D_Docs folder.

Conceptually, there are several groups of tables within the PCMMR schema, which itself lives within the larger CISS database and has foreign keys into those CISS and sdsadm tables.

- The “lookup” tables, starting by convention with the prefix “PCM_STD_”, contain fixed reference values which rarely change, and are generally considered read-only within the application context.
- The audit history tables ending in “_H” are populated solely via internal database triggers each time a row in the corresponding table is inserted, updated or modified. For example the PCMM.STAFF_POSITION_ASSIGNMENT table has a corresponding PCMM.STAFF_POSITION_ASSIGNMENT_H table which can be used for auditing purposes later to recreate who made what changes at what times.
- The main domain tables representing the primary business domain objects, such as Teams, Positions, Staff and Assignments. These tables form the primary set of data within the PCMMR database.

3.2.2. Database Information

Table 14 Database Inventory

Database Name	Description	Type	Steward
Legacy PCMM VistA files	FileMan files, located within each VistA site, accessed via specific APIs and RPCs. These files contain all existing patient, provider, team and position information for PCMMR.	Legacy data store	VistA site
CDW PCMM consolidated data store	The CDW contains an aggregation of legacy data across all VistA sites which is currently used for analysis and regional/national reporting.	Legacy data store	CDW team
CISS	This database is shared by the main CISS framework, the OHRS partner application which uses the “OHRS” schema, as well as the “PCMMR” partner application which uses the “PCMMR” schema. It also includes standard data in the “sdsadm” and standard	Replace old separate PCMM data stores	Partnership between CISS, OHRS, and PCMMR teams (and possibly other future partner applications)

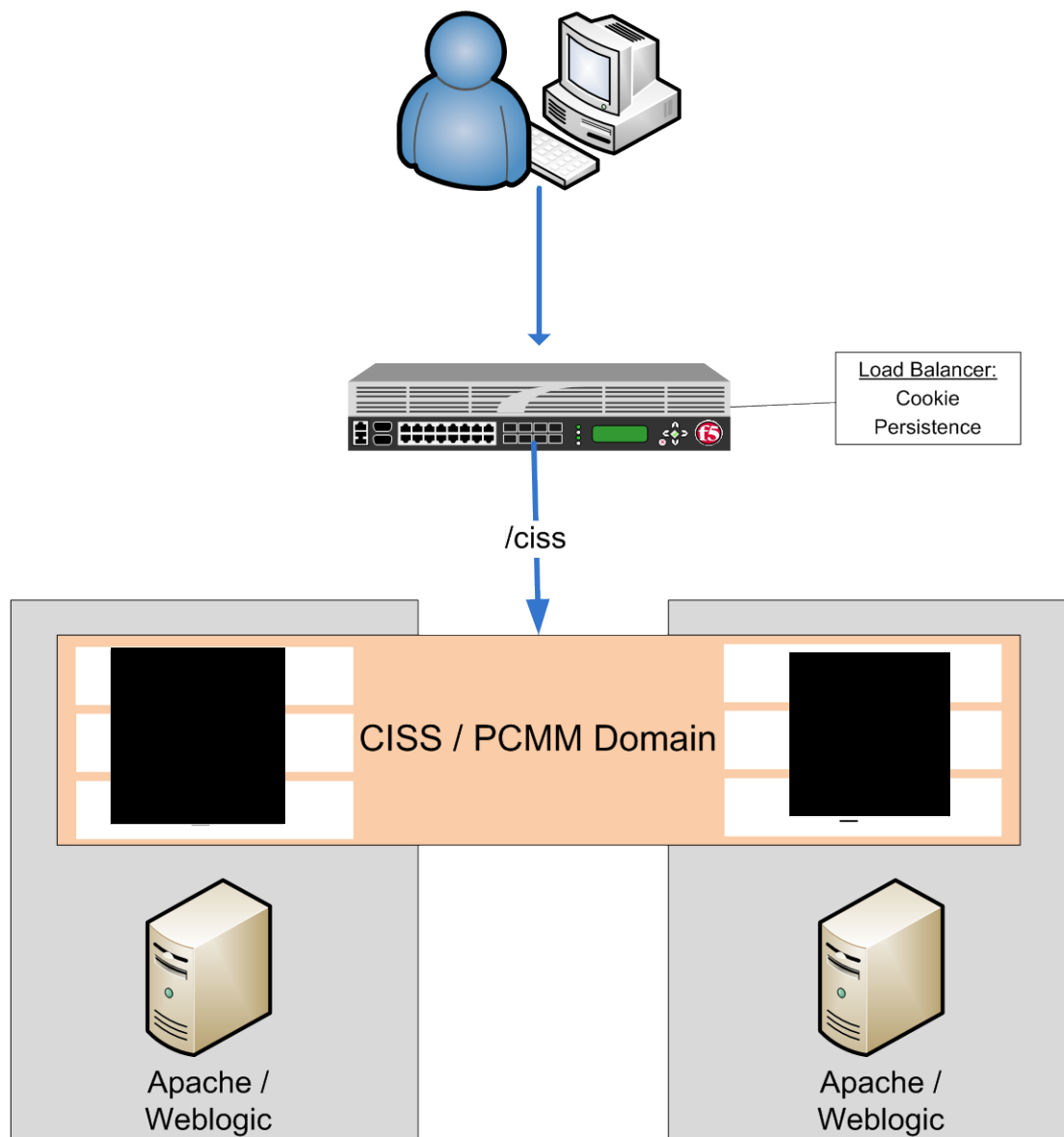
This context diagram identifies specific enterprise services and applications with which CISS and PCMMR will interface. CISS will be integrated in the current VHA infrastructure and contain interfaces to various enterprise services identified in the following table.

Service	Category	Integration Technology	Notes
VistA	Enterprise	VistaLink	
Active Directory	Enterprise	Spring LDAP	Authentication and Authorization
CISS Database	Internal	Java Persistence Application Programming Interface (API) (JPA)	Database for CISS and PCMMR
PCMMR	Partner Application	Java Portlet (via WSRP local proxy)	
CMS	Internal	Content Management System provided by WebLogic Portal Server	Provides dynamically generated content to CISS landing page and homepage

Table 2 – Enterprise Interfaces

3.3.1. System Criticality and High Availability

Consider the following deployment structure for PCMMR:



PCMMR has multiple Weblogic instances (potentially running in different VMs on the same or different physical servers) that can be stopped and started independently. Each managed server (gray box) contains a complete PCMMR runtime which works to service a customer's web session. Once a customer logs in, they are associated with one of the runtimes and remain associated to that server until they log out. When one of the managed servers goes offline, the hardware load balancer at the top automatically routes all user requests to another active runtime.

To ensure high availability, hardware upgrades can be performed on one inactive runtime while the other continues to run and service customer requests. To ensure active users don't have their sessions disrupted,

the load balancer can be reconfigured on the fly to direct customer traffic to all but one server, and the hardware teams can wait until all users sessions have completed before taking the server offline for upgrades.

3.3.2. Special Technology

Table 15 Special Technology Requirements

Special Technology	Description	Notional Location	TRM Status
Load Balancer	A distribution technology that allows user requests to be sent to working pieces of the application deployment	The user requests (coming from web browsers) will be sent directly to the load balancer.	Yes

3.3.3. Technology Locations

Table 16 Technology Location

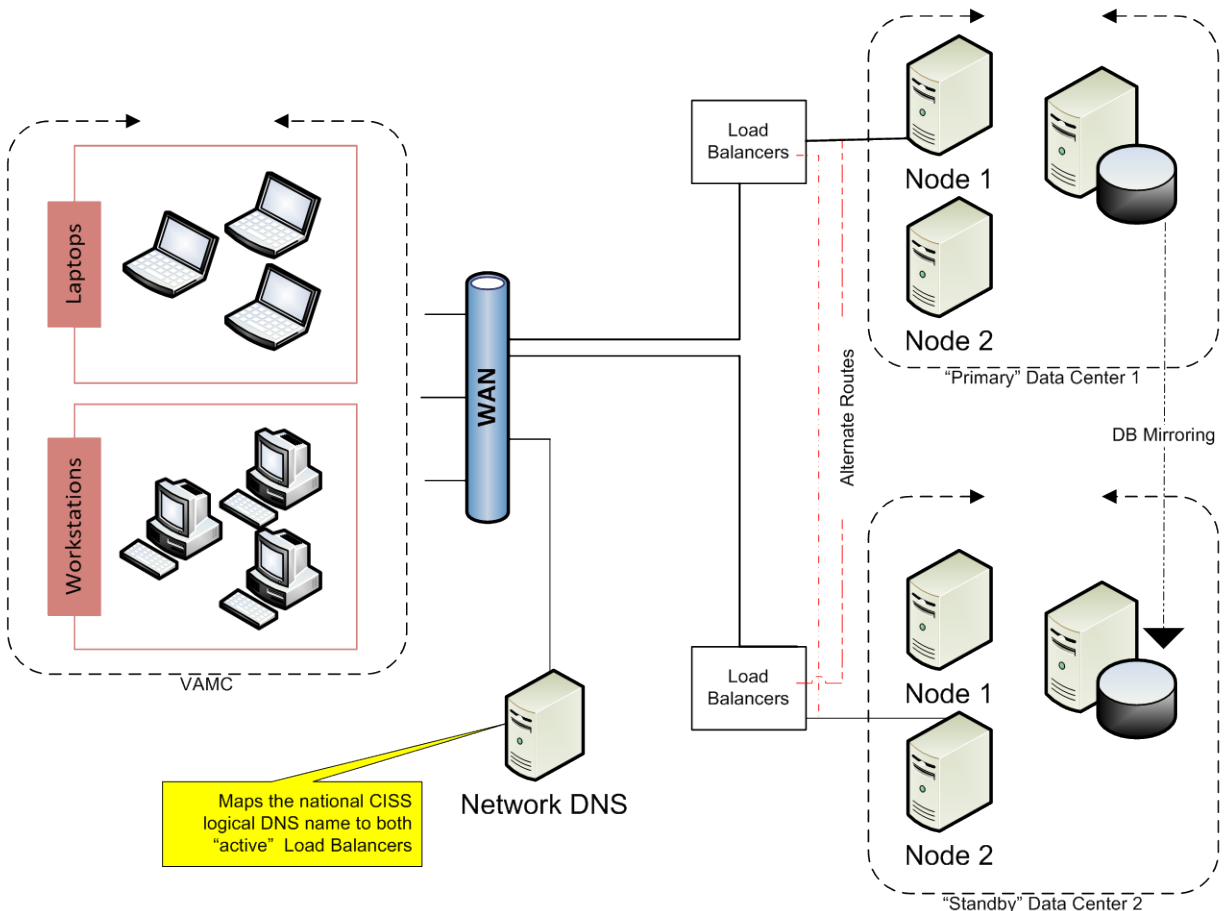
Technology Component	Location	Usage
Production 1		
Workstations	N/A	
Special Hardware	N/A	
Interface Processors	Embedded within application code	Connectivity to Vista instances and external application instances
Legacy Mainframe	Existing VistA sites	Query for existing patients and staff; synchronize data with external systems such as CPRS and MyHealtheVet
Legacy Application Server	N/A	No host application server is necessary to run the legacy PCMM software
Legacy Databases	Internal VistA files (see Legacy Mainframe above)	Storage of all VistA data
Source Code Repository	HP–cloud hosted development Subversion server and Rational ClearCase server on VA network	Daily code checkins and builds in SVN; periodic stable checkins to ClearCase to meet VA requirements
Production 2		
	ESX Virtual machine setup at Site B	Disaster Recovery environment
Certification		
	Virtual Machine(s) on VA network	Integration and additional development on VA network to VA enterprise systems
SQA		
	Virtual Machine(s) on VA network	Internal functional, automated and smoke testing of application during development; demo to users

		of new functionality
UAT		
	Virtual Machine(s) on VA network	Stable test environment updated once per sprint for demo purposes to the VA business customer and SMEs.
Development		
	Virtual Machine(s) on VA network	Integration environment for development team (pre-SQA)

3.3.4. Conceptual Infrastructure Diagram

3.3.4.1 Location of Environments and External Interfaces

Figure 5 CISS network overview



3.3.4.2 Conceptual Production String Diagram

The PCMMR application will be deployed into one of the represented “CISS App Servers” above and communicate with the CISS National Database where its data is stored.

4. System Architecture

4.1. Hardware Architecture

The production PCMMR will be deployed internally in a VA-approved Red Hat Enterprise Linux virtual machine and communicate with the existing CISS Portal Linux virtual machine over the VA network. The virtual machine itself may be abstracted across several physical devices, such that the Virtual Machine

host software can replicate and provide seamless redundancy in the event that one physical device fails. Database Storage is on an existing Storage Area Network (SAN) located in the Martinsburg, WV data center, which has its own set of failsafe tools. Each physical device will have multiple NICs (network interfaces) to provide uninterrupted network connectivity, as well as redundant power supplies for the same reason.

4.2. Software Architecture

4.2.1. CISS

The architectural goal of CISS is to provide a shared platform for the business applications of its customers. Each business application that leverages CISS is called a “partner application” and forms a partnership with the CISS set of services and portal infrastructure. PCMMR is an example of a partner application, although one of its architectural goals is to be independent of CISS services and simply rely on CISS as a portal wrapper and authentication provider.

4.2.2. Partner Applications

Each partner application must comply with and determine how it will implement the following key items:

- Integration technique (WSRP, URL)
- Local partner application or Remote partner application
- Throughput forecast for a three-year period
- Context sharing requirements
- Application role-based exposure
- Data storage location (for the PCMMR logical / physical data models, see section 5)
- Multiple authentication reduction
- COOP strategy (to include WAN failure)

The partner applications are notified of these requirements during development by the CISS team.

4.2.3. Code Framework

CISS provides a set of reusable components and services that any CISS application can leverage. These services are discussed in the Process View.

4.3. Software Architecture – PCMMR

The primary project goal of PCMMR is to reengineer the legacy PCMM application such that all use cases and user requirements are satisfied for each deliverable deadline: six-month initial prototype, initial operating capacity (IOC) and final operating capability as PCMMR moves into national production release. The architectural and design goals for PCMMR are to build a product meeting these project requirements using only the most modern, cutting-edge tools and frameworks, and to do so using the least amount of overhead/“boilerplate” code possible. The final PCMMR product will provide a dynamic web-based user interface, integrate seamlessly with CISS, leverage existing VistA and other enterprise

connectivity where needed, and provide privacy controls on top of its design for clinical teaming information.

During the process of reengineering PCMM into a single enterprise application, the new architecture must ensure that all existing systems that exchange data with legacy PCMM are not abandoned. The new application will support exposing its internal data as web services, which can then be consumed by other approved applications. However, the PCMMR team can't expect all dependent systems to immediately upgrade their software. To provide a manageable transformation of those systems, PCMMR will synchronize necessary portions of its internal data model with legacy VistA files. This will trigger the same events, messaging and system-to-system data exchanges as in the legacy PCMM. For more details of this process see section 6.2.

4.3.1. CISS Compared and Contrasted

The PCMMR minimalistic approach described above can be contrasted to that of CISS, whose job is to function as a portal framework and service provider. Whereas CISS may have more than one way of doing something – for example, providing data via web services in addition to the normal database approach, or implementing CCOW session sharing in addition to local and federated portlet sessions – PCMMR strives for a lean and clean design. It will carve out only what functionality is necessary from CISS while maintaining its own internal set of simple domain objects and services. CISS was developed historically using large XML files to initialize tools like the Spring framework and Hibernate ORM; PCMMR will use the more streamlined Annotation-based approach. PCMMR will adhere to the idea of “convention over configuration” when implementing several application layers, including Spring autowiring, the Spring Portlet MVC presentation layer, a transactionalized service layer and JPA/Hibernate-based data layer. The well-documented behavior of these tools and their adherence to & implementation of public Java Specification Requests such as 250 and 330 (for annotations) and 286 (for portal/portlet functions) will minimize the ramp-up time of future developers supporting PCMMR.

4.3.2. Standard Partner-System Implementation

As described in section 4.2.2, PCMMR will use the following values when serving as a partner application to CISS:

- Local portlet application & Integration technique (WSRP, URL)
 - PCMMR will serve as a local portlet to the main CISS portal, and thus will communicate to and from the parent portal using WSRP 2.0. This includes WSRP eventing and advanced WSRP features, as described below in the context-sharing section. The WebLogic-proprietary “local proxy” flag will be enabled, removing the need for TCP socket communication between the portal and local portlet.
- Throughput forecast for three-year period
 - Throughput expectations for PCMMR are outlined in the PWS and other project documentation found on the PCMMR SharePoint site. In general, PCMMR will respond to all user-interface functions within a few seconds of their submission, and will be able to serve 1000-2000 simultaneous user sessions. Longer running processes such as reporting functionality or batch processes will be implemented in an asynchronous manner separate from the main PCMMR application and will notify/“call back” the main application & user when complete.
- Context sharing requirements

- PCMMR not only functions as a portlet application for the main CISS site, but also uses WSRP eventing to synchronize context data to and from the main CISS site. This context data includes (but is not limited to) items like the logged-in user profile, roles and permissions, currently selected duty station and currently selected patient⁸.
- Application exposure
 - The PCMMR application inherits the granular authentication and authorization functionality provided by CISS. Although any VA user may log into the CISS parent portal, each user must be explicitly assigned a role which itself is granted the permission to access PCMMR. Furthermore, all PCMMR functions have associated permissions which must be assigned to the user's role in order for the user to be able to activate that portion of the application.
 - PCMMR created a separate user editor administration portlet to manage its local set of database users and their roles in the system. This editor will synchronize the roles assigned to users with the VA LDAP server. PCMMR will not interface with Identity and Access Management (IAM) for system user management, but will rely on CISS for authentication / authorization until a longer-term SSO solution is provided by VA.
- Data storage location
 - PCMMR shares a common single database with CISS. This provides the ability to have PCMMR tables directly reference other CISS tables (and also third-party tables such as the SDS or STS data sets), providing a foreign key constraint guarantee to that data.
- Multiple authentication reduction
 - PCMMR does not currently need authorization or authentication for any of its functions outside of that provided by the main CISS portal. The logged-in user context is shared by PCMMR and other partner applications as described above. If future connections need to be established to other enterprise systems which have additional authentication requirements (such as Vista, which uses access/verify codes), it will be implemented separately by PCMMR.
- COOP strategy (to include WAN failure)
 - PCMMR depends completely on the CISS portal to provide an interface to the user. If the CISS portal system is stopped or becomes inaccessible, all PCMMR functionality will be unavailable. CISS is implemented in production using a distributed and clustered approach and also has a fully implemented disaster recovery (DR) plan in case of a long-term outage. Final production installation of PCMMR will include the same guarantees such that its content will be replicated and provided in case of any singular server failure.

PCMMR uses a variety of enterprise tools. While the code is primarily written in Java, other tools are also used: JSP / JSTL, JPQL, HTML, Javascript, and XML. It is closely aligned to the Spring framework for dependency injection / autowiring, transaction support and abstractions on top of a variety of other tools, including the JSR286 portlet standard.

⁸ Future functionality

The code is organized into four main sections:

- **Model** – Contains the business model classes in the application representing the various domain entities in PCMMR. This layer is simple and makes use of a hierarchy of persistence superclasses that ensure all persistent objects share a common set of traits. These traits include, but are not limited to:
 - A unique integer ID number, used in the database as the primary key
 - A version number used by the Persistence layer tool to ensure updates occur serially and don't collide with one another
 - A set of audit columns, capturing the creator, date/time and timezone for the object
 - A set of audit columns capturing the most recent updater, date/time and timezone of the object

The model classes primarily contain simple functionality and references to one another for small helper methods.

- **Persistence** – Contains the data access classes which abstract storage, update and retrieval of objects in various data stores. These stores can include the primary database, LDAP, web services, JMS servers and other repositories of data inside and outside of the application. The data-access-object (DAO) pattern is used heavily as well as a set of superclasses which integrate with the JPA standard to provide database persistence. The data-transfer-object pattern is discouraged and not used by PCMMR; instead, the domain objects themselves are directly persisted to the data stores.

To implement a new DAO, it's best to copy both an existing DAO interface class (located in the `gov.va.med.pcmm.persistence.dao` package) and that DAO's implementation class (located in the `gov.va.med.pcmm.persistence.impl` package). Because of the `@Controller` annotation, Spring will automatically discover your new DAO and inject it anywhere else it's needed; it's a good idea to add it to the common set in both `AbstractPCMMController` and `AbstractPCMMServiceImpl`. Generally all DAOs are descendants of `GenericHibernateDaoImpl`, which provides all the common functionality: finding by primary key, saving, deleting, and the ability to run arbitrary JPA queries. If the DAO is for a class that implements `Comparable`, then you should instead extend `GenericHibernateSortedDAOImpl`, which provides method that return `SortedSets` in addition to `Lists`. Finally, if your DAO is for a lookup class (mapped to one of the `pcmm.pcm_std_*` tables), your DAO should extend `GenericHibernateLookupDAOImpl` which adds methods to retrieve the lookups by code or name.

- **Service** – Contains classes which implement the business logic and validations required for the operation of PCMMR. All classes in this layer are automatically transactionalized by Spring (a VA-approved framework – see TRM section 1.7.1) using aspect-oriented programming (AOP) techniques. All methods are implemented as an atomic unit of work and must either complete or fail as a whole. This ensures the database is always kept in a predictable, consistent state.

Like the other layers of the application, a helpful set of superclasses provides this transaction configuration automatically, as well as provides many other resources that may be helpful (e.g. instances of the DAO classes). To implement a new Service class, it's best to copy an existing Service interface (located in the `gov.va.med.pcmm.service` class) and that Service's implementation class (located in the `gov.va.med.pcmm.service.impl` package). Because of the `@Service` stereotype annotation, Spring will automatically discover your new Service and inject it

anywhere else it's needed; it's a good idea to add it to the common set of Services in AbstractPCMMController. Generally, all services are descendants of AbstractPCMMServiceImpl, which provides access to all DAOs, links to a few CISS services and the shared configuration for transactions and transaction rollbacks.

- Web – Contains classes that primarily utilize the Spring lightweight Model-View-Controller pattern (with a preference given to annotations over a separate XML configuration). This pattern defines:
 - Controller classes – The function of these classes is to handle all logic surrounding the interaction with requests coming in from users via their web browsers. Saving items to/from the various JEE contexts (Session, Application, Request), providing reference data and managing request parameters are all handled by the controller classes. The controllers generally will construct a Model, which consists of both reference data and also a Command object (defined below) and send the model to a View (also defined below) for rendering.
 - Command objects – These are often simple Plain-Old-Java-Objects (POJOs) that are bound via Spring automatically to form input values in the web browsers. As users submit forms or generate requests, the values associated with those requests are stored temporarily into a Command object instance so that Spring can manage the set of values as a unit and associate errors, if needed, to specific values. The command classes are accessible from all controller methods, and often the controllers have logic which manipulates those values as the conversation with the user occurs.
 - Views – Once the controllers have built the Model, they forward the request to a View. Spring contains an abstraction for a variety of View tools and technologies; PCMMR uses the standard JSTL view, which allows for the model to be rendered in a standard JSP page using JSTL tags and the JSP Expression Language (EL). These JSP views also contain Javascript which runs on the client side in the web browser.

Additionally, a fifth package “Util” is for static utility classes whose job is just to provide helper methods to the main four sections above.

4.3.3. AJAX

The PCMMR team has implemented AJAX communication between the user web browser session and the PCMMR server application, in addition to the normal Web layer above. Following Spring's conventions and the JSR-286 Resource Request pattern for implementing AJAX requests, Javascript in the client-side web browser can send asynchronous XML requests directly into the Spring controllers described above, and those controllers can respond with data values serialized into JSON format. All of this happens transparently to the developer, and a variety of abstractions are put in place such that the developer can annotate which methods are AJAX methods and which are normal Spring MVC-style methods. Spring uses the Jackson JSON encoding tool to serialize the annotated domain objects into JSON before sending back to the web browser.

4.4. Communications Architecture

The following items represent the different communication paths of PCMMR:

- **CISS Portal Server -> PCMMR Portlet (WSRP)** – Although PCMMR has the functionality and look-and-feel of a complete standalone application, it technically still functions as a portlet, which by definition is but a single contributor among many when rendering the final content for

the end user. The CISS framework therefore needs to request content from the PCMMR portlet application, and send incoming user parameter and form submissions to the PCMMR portlet. This is done over WSRP-encoded HTTP via a proprietary WebLogic-hosted local proxy connection between the CISS portal application and the PCMMR portlet (running on the same machine and inside the same WebLogic domain).

- **Attended Server -> JMS -> Unattended server** – This is a common path originally implemented by the CISS framework and reused by the PCMMR application to support asynchronous request handling. These async requests include both report requests and batch processes which are generally long-running processes.

The Attended server and Unattended servers are both identical instances of the same PCMMR application, but configured via properties slightly differently. Users interact with the attended server via their web browser, and the attended server sends requests to, and relies upon, the Unattended server to process long-running tasks. The means of communication is provided by a JMS queue (hosted by the parent WebLogic application server), and messages that are added to this queue automatically activate a message-driven bean (MDB) on the Unattended server side. The JMS queues are distributed Store-And-Forward queues as provided by WebLogic.

- **Web Services** – PCMMR will support exposing a variety of its service-layer methods via web services such that legacy systems can retrieve information about patient teams, staff member assignments, etc.
- **HL7 message stream from MVI** – PCMMR will support processing of incoming messages from the Master Veteran Index (MVI). These messages contain notifications of patient identity trait updates, merges and splits across different patients. As the PCMMR local data is updated based on each message, an acknowledgement will be sent back notifying MVI that PCMMR was updated.

PCMMR also sends HL7v3 (SOAP webservice) requests to MVI for access to the patient primary view and to initially register PCMMR as a consumer of future primary view change messages (HL7v2).

5. Data Design

The data for PCMMR is stored exclusively in a dedicated schema inside the existing shared CISS SQL Server database. The login to this database by the PCMMR application provides the PCMMR code the ability to reference not only the PCMMR schema tables, but also the shared CISS tables, as well as other reference data stored in the CISS database, such as SDS data and STS.

The latest schema for the database is stored in the following PDF:

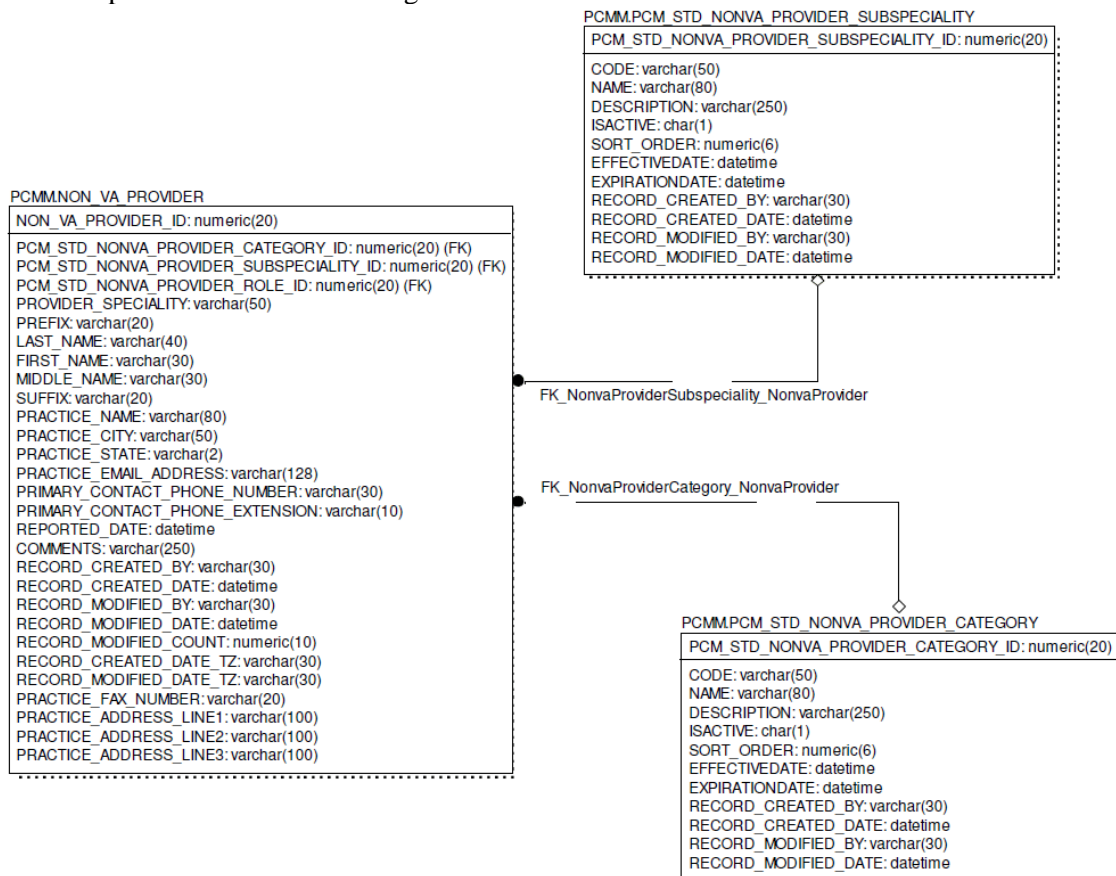


5.1. Database Management System Files

The database is designed as a set of normalized tables, which are closely related to (but may not match exactly) the business domain objects defined by the Java code. These tables can be logically split into two sets:

- Reference / “Lookup” data – This data exists mostly as a set of standalone reference information which is used by other objects in the system for configuration and as attributes. For example, the fact that some objects are considered “Active” and others “Inactive” is implemented in the database by those objects referencing a lookup table called “STATUS”, which contains one row for active and another for inactive. Some of these reference tables may be updated by the administrative users of PCMMR during the application runtime, but these updates will occur much less frequently than updates to the primary data objects defined below.

An example of reference table usage is:



As shown, the two tables on the right are lookup tables containing the subspecialties and categories for Non-VA providers. The primary table on the left has a foreign key relationship to specific rows in each lookup table. Notice that the lookup table columns are identical; this consistency is shared by all lookup tables in the PCMMR schema.

Code	Unique code for this lookup value among the other set of lookup values in this table (but not
------	-----------------------------------------------------------------------------------------------

	across tables). This is generally either a number or an all-caps value separated with underscores. No spaces are allowed.
Name	Short description of value. If a sort order is not defined (null), then the lookup values are sorted by this column by default.
Description	Longer explanation of value
Sort Order	Integer sort order; if non-null, these values override the default alphabetical sort order by Name
Effective Date	The activation date for this value
Expiration Date	The inactivation date for this value

- **Primary Objects** – This data represents the main fundamental domain objects used by PCMMR to contain the main business data. An example of a primary object is the Team object, containing a set of attributes for itself (such as the Team Role, the description and capacity count) as well as links to other primary objects (such as Staff Member Assignments).

An example of a primary object is the PCMM.NON_VA_PROVIDER table in the above diagram. A subset of the columns listed are:

NON_VA_PROVIDER_ID	Primary key unique integer identifier. The naming convention for all primary key columns is the name of the table, followed by “_ID”.
PCM_STD_NONVA_PROVIDER_CATEGORY_ID	A foreign key into the category lookup table described above. Foreign key naming conventions are to have the foreign key column named the same as the primary key in the foreign table being referenced.
FIRST_NAME	A normal data column containing an attribute of the object being represented.
RECORD_CREATED_BY / RECORD_CREATED_DATE	The system user who initially created this record, and the time of creation. These values don't change as the record is updated.
RECORD_MODIFIED_BY /	The system user who has most recently updated the row, and the date of the most

RECORD_MODIFIED_DATE	recent update.
RECORD_MODIFIED_COUNT	This is an integer representing the version of this row in the database. As the row is updated, this number is incremented by 1, which prevents two users from updating the same row twice at the same time.

An analysis was done by the development team to determine which reference data items would be feasible to update and what the implications might be for the system; see the following embedded document for that analysis:



The access methods to the database from the PCMMR application is JDBC.

Some features of the SQL Server database are discouraged by the VA customer, including stored procedures. These items are implemented explicitly in the service layer Java code of the PCMMR application.

The scope of the number of transactions and concurrent users of the database is proportional to both the number of users concurrently using PCMMR as well as the number of incoming requests from external systems - CPRS popup window views, MVI patient updates, background batch job executions, etc. The PCMMR application implements several strategies for in-memory caching to reduce the load on the database. Certain lookup values are annotated as “eternal” caching, meaning they are permanently stored in memory and only retrieved once upon application startup.

5.1.1.JPA mapping example

In order to link a business domain object in PCMMR to a database table, the object relational mapping (ORM) settings must be updated. These settings describe how to map the Java fields and relationships (object graph) to the relational set of tables in the database. The VA-approved J2EE standard for this mapping is using the Java Persistence Architecture (JPA) standard. Although there is more than one way to configure JPA, the PCMMR standard is to use annotations within each Java class. The advantage of this is to keep the definition of each field and its configuration nearby, so that developers don’t need to track down a separate configuration file. The following example shows how to map a Java object to a table:

```
@Entity
@Table(name = "MODEL_TEAM_POSITION", schema = "PCMM")
@AttributeOverrides({ @AttributeOverride(name = "id", column = @Column(name = "MODEL_TEAM_POSITION_ID")) })
public class ModelTeamPosition extends AbstractPCMPersistentWithVersion<ModelTeamPosition> {

    // ----- Fields

    private TeamRole teamRole;
```

```

private boolean required;

private ModelTeam modelTeam;

// ----- Common Methods

@Override
public int compareTo(ModelTeamPosition u) {
    if (equals(u))
        return 0;

    return new CompareToBuilder().append(getTeamRole(), u.getTeamRole())
        .toComparison() > 0 ? 1 : -1;
}

@Override
protected boolean requiredEquals(ModelTeamPosition u) {
    return new EqualsBuilder().append(getTeamRole(), u.getTeamRole())
        .isEquals();
}

@Override
protected int requiredHashCode() {
    return new HashCodeBuilder().append(getTeamRole()).toHashCode();
}

// ----- Accessor Methods

@Type(type = "yes_no")
@Column(name = "REQUIRED_IND")
public boolean isRequired() {
    return required;
}

public void setRequired(boolean required) {
    this.required = required;
}

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "PCM_STD_TEAM_ROLE_ID")
public TeamRole getTeamRole() {
    return teamRole;
}

```

...

The items of interest above are:

Item	Description	Notes
@Entity annotation	Defines this object as a persistent Entity, per JPA standard	Superclasses to this class must be marked as @MappedSuperclass if those classes also have persistent fields which are inherited by this class
@Table	Defines the table used	

	to contain instances of this object	
@AttributeOverrides	A collection of overrides to superclass persistent definitions	For PCMMR, this is generally used to override the default “id” column to reflect the name of the table’s primary key column.
extends AbstractPCMMPersistentWithVersion	Ancestor of all persistent PCMMR classes	There is a hierarchy of superclasses which all persistent classes can inherit from, depending on the function of the PCMMR class
@Override protected boolean requiredEquals / @Override protected int requiredHashCode	Required implementation of equals and hashCode	JPA requires that classes implement equals and hashCode consistently (using the same set of fields and in the same order) using “business key equality”, meaning fields other than the ID are used for comparison. Some common functionality for these methods is implemented in the parent AbstractPCMMRPersistent.
@Type(type = "yes_no")	Type specifier for a particular field	JPA/Hibernate specifies particular data types for each field; if not specified, the String is the default type
@Column(name = "REQUIRED_IND")	Column specifier for a particular field	Attached to the getter method, along with @Type, this defines the name of the column, it’s nullability and several other attributes.

6. Detailed Design

6.1. Software Detailed Design

6.1.1. Module “CRUD editors”

To implement a CRUD editor for a simple object in PCMMR, the following components must be developed:

- **Domain objects** representing the main object, its attributes, and any children objects it references. These objects will extend the appropriate parent class which provides a common set of persistence attributes and enforces some necessary methods to be implemented (such as requiredEquals() and requiredHashCode() for Hibernate purposes). Within these objects,

annotations need to be added which define the database table and columns that link to each of the object's fields, as defined by the JPA 2.0 standard. Additionally, annotations can be added to the fields which support simple JSR-303 validations on the fields (such as size or length restrictions, minimum or maximum values, and some formats like email addresses). Finally, serialization annotations can be added which define the set of fields that are serialized automatically by the Spring / Jackson subsystem when sending this object to the client web browser via AJAX.

- **Database tables** that contain all columns and foreign relationships necessary to persist the domain objects
- **Persistence layer classes** (DAOs) which provide CRUD methods for the new object being saved to or retrieved from the database. A common set of DAO superclasses exist which should be extended to provide almost all basic CRUD functionality using the JPA 2.0 techniques. A specific superclass exists for Lookup classes which share a common set of fields (such as name, description and sort order). Finally, a custom superclass exists for all domain objects implementing the Java Comparable interface, such that they can be returned as a SortedSet automatically.
- **Service-layer transactional classes** that define the main functions (save, update, delete) and also perform business-level validations for the persistence or modification of the domain objects. These methods represent the main application logic, and should not depend on any web-layer classes. By extending a common service parent class, they automatically inherit all DAOs for access to the database, as well as the @Transactional attribute provided by Spring which automatically creates new database transactions before the method begins, and closes (or rolls back) the database transaction after the method exits. It's good practice to perform some common validation of the input parameters inside the method, and to throw a ServiceValidationException (a custom PCMMR type) if any business logic validations fail. Finally, these methods should be restricted to users with certain permissions, via the @PreAuthorize Spring Security annotation.
- **Web-layer model-view-controller classes** which manage the user interaction with forms that display and save the object. As described elsewhere, these classes consist of one or more Controller classes (annotated with @Controller), one or more "command" classes which are simple POJOs which map directly to form input fields, and JSP files containing markup and JSTL tags which render the HTML forms and perform client-side processing and validation. Additionally, the JSP files make extensive use of the jQuery Javascript framework for standardization and simplification on the client side.

6.1.1.1. Processing

The flow for a user session and the respective events in the CRUD editor code is:

User Action	PCMMR processing events
User clicks View object link, which was generated on the server side via the Portlet 2.0 standard. Embedded in the link is the ID of the	<ul style="list-style-type: none"> • PCMMR inspects URL and based on the @RequestMapping annotations in the @Controller classes, decides which method to activate. • PCMMR activates the "view [object]" controller method, injecting all parameters as required, and expecting a model and view to be returned. Often the developer will create a Command object and save it in the model which backs the form shown to the user. This command object is also

object requested.	<p>specifically bound to (and retrieved later from) the portlet session, such that when the user submits the form, the values remain available in the event of a validation error.</p> <ul style="list-style-type: none"> • The view returned maps to a specific Tiles definition; Spring delegates rendering of the view to the Tiles subsystem • Tiles assembles all JSP files as configured in pcmm-tiles.xml and generates the final markup. The JSP files used can contain JSTL tags, custom PCMMR tags, Spring security tags and Tiles tags. Final markup is sent back to the user for display. This happens during the Portlet “Render” phase – see the portlet specification for details.
User makes changes to some form input values and clicks “Submit”.	<ul style="list-style-type: none"> • Spring inspects the URL and parameters being submitted and activates the “save [object]” controller method. This method is activated during the Portlet’s “Action” phase (although that may not be immediately obvious based on the method annotations – see the portlet specification for details). If a command object was annotated such that it is saved in the user’s web session, it is retrieved from the session and all the submitted form values are automatically bound to the command object’s fields. • The controller method then performs some simple validations (or delegates these checks to a separate Validator class), then calls the service method whose job it is to save or update the object in the database. The service method then performs its business-level validations and logic, calling the persistence-layer DAO class if everything checked out, and the object is saved to the DB. • Once the service method returns with the new (or updated) persistent object, it sets the appropriate render parameter such that Spring will activate the correct success page during the upcoming render phase. • If any errors occurred in the service method, the transaction is rolled back, the database is left unmodified and the user is often returned to the same form page so he/she can deal with the validation errors.

Obviously the detailed behavior for any particular object has additional customizations, but the general workflow above is shared by all CRUD editors. A solid understanding of the complete PCMMR tool stack (namely the Spring Framework, JPA/Hibernate, JSPs) is needed to implement a fully-functioning CRUD editor.

6.1.2. Module “Search Screens”

To implement a standard (non-popup) search screen in PCMMR, the following components must be developed:

- **Domain objects** – generally these will already have been developed in preparation for the CRUD screens above. If not, consider making the CRUD screens first since these will usually help finalize the object attributes, and the search criteria used in the search screens will be some subset of those attributes.

- **Database tables** containing the domain objects will generally also have been developed during the CRUD editor phase; if not, the domain objects, JPA mappings and database tables are required before search screens can be developed.
- **Persistence layer classes** (DAOs) which provide search methods for the new object. The recommended approach is to have a single method called `findByCriteria()` which accepts several nullable parameters representing various ways of restricting the final result list. For all non-null parameters, a JPA query is built with the applicable restrictions and a result set is returned. In the event that the object implements `Comparable`, the developer might choose to return a `SortedSet` as compared to a simple `List`, for convenience.

It is considered an antipattern to have any “built-in” restrictions in the JPA query, even if these seem common-sense, since they introduce implied dependencies in the code. For instance, restricting the set of objects returned to those within the user’s duty station is discouraged; instead, a parameter specifying the station ID (or set of applicable station IDs) should be provided by the caller.

Another pitfall to avoid is joining children objects from the main object if they are not really needed, given the set of non-null parameters. These can slow down the query since they create unnecessary joins at the database level. Be sure to only join those children objects or external referenced objects when absolutely necessary.

- **Service-layer transactional classes** are generally not needed since the web layer simply calls the DAO methods directly for the list of matching objects.
- **Web-layer model-view-controller classes** are used to model the search criteria and contain the `List` (or `SortedSet`) of matching results from the database. These classes call the DAO `findByCriteria` method and display the results in a standardized markup table format.

6.1.2.1. Processing

The flow for a user session and the respective events in the search screen code is:

User Action	PCMMR processing events
User clicks Search for Object link, which was generated on the server side via the Portlet 2.0 standard. Any necessary parameters further defining the search should be passed to the render method.	<ul style="list-style-type: none"> • The same Spring workflow events occur as in the CRUD editor display screen section above. What changes for the PCMMR search screens is that the command object instead contains a separate field for each search field, and a <code>List</code> (or <code>SortedSet</code> – developer choice) of search results, which is initially empty.
User makes changes to some form search values and clicks “Submit”.	<ul style="list-style-type: none"> • The same spring workflow events occur as in the form submission section of the CRUD editor description above. Spring binds the search parameters to the command object, and in the action handler method (analogous to the <code>POST</code> method in the Servlet world), the controller calls the <code>findByCriteria</code> DAO method and sets the list of results in the command object. • The JSP can check the size of the result list and, if non-empty, display all search results in a standardized table below the

6.1.3. Module “AJAX methods & Dialogs”

AJAX stands for Asynchronous Javascript and XML, and represents a way for applications to exchange data asynchronously with the server (that is, outside of the standard browser request/response processes). Implementing AJAX functionality in PCMMR is straightforward, due to the Spring Framework’s flexible implementation of this protocol. To call a server method and process its response, the following components must be developed:

- **Server Side** – In any controller class, implementing a method that is activated for some AJAX request is simple. Observe the following method signature:

```
@RequestMapping("find-rooms")
@PreAuthorize("hasRole('\" + CISSPermissionType.READ_TEAM + '\")")
public @ResponseBody List<Room> findRooms(
    @RequestParam String number,
    @RequestParam(required = false) String onlyActive,
    ResourceRequest request, ResourceResponse response)
```

Note that the ResourceMapping annotation defined a unique string, just like all other controller methods. Also these methods share common authorization annotation @PreAuthorize with other controller methods. The change is the @ResponseBody annotation, which notifies Spring that the method will be generating the body of the response directly, and that Spring should not attempt to delegate rendering to a View. The method can accept any @RequestParam-annotated parameters and can return any JSON-serializable Object (POJOs, Collections, etc) or primitive⁹.

In coordination with the Portlet 2.0 standard, Spring processes AJAX requests via the ResourceRequest portlet subprocess, which is why the developer can optionally include the ResourceRequest and/or ResourceResponse objects in the method signature.

- **Client Side** – Generally in AJAX-powered applications, a Javascript framework is used to simplify submission of Asynchronous XML messages to the server and process the results returned. PCMMR is no different and relies extensively on the jQuery open source Javascript framework. Observe the following snippet:

```
<portlet:resourceURL var="getMultiPCPURL" id="get-multipcp" />
```

...

```
function AJAXExample(objectId, theCallback) {
    $.AJAX({
        url : '${getMultiPCPURL}',
        dataType : 'json',
        data: {
            multiPCPId: objectId
        },
    },
```

⁹ With some restrictions - see the Spring/Jackson JSON integration documentation for details

```

        success : function(response) {
            var r = response.AJAXResult
            theCallback(r.multiPCP)
        }
    }).error(function(e) {
        alert("Couldn't get the MultiPCP object: " + e)
    })
}

```

This Javascript method shows a simple way to submit an AJAX request to the server and asynchronously process the results. The URL is generated at the top of the JSP and contains the same Spring “get-multipcp” as what’s defined in the Controller @RequestMapping. In the Javascript function, we call @.AJAX jQuery method, which activates the process described below.

6.1.3.1. Processing

The flow for a user session and the respective events in the search screen code is:

User Action	PCMMR processing events
Some client-side event triggers the method above – for example, it could be clicking a button, expanding a show/hide table, or any Javascript that needs to communicate to the server.	<ul style="list-style-type: none"> The Javascript method uses jQuery’s .AJAX() method, which constructs an asynchronous XML (POST) request behind the scenes and includes all the parameters defined in the data { ... } section. The controller method on the server is activated (which may be short or long running). The expected response format (“JSON”) is automatically sent by the web browser to the server, which notifies Spring to automatically encode any results from the controller method as JSON via the Jackson tool. Meanwhile, since AJAX is asynchronous, the Javascript method proceeds as usual on the client (in this case, exiting the method). On the server, once the controller method has completed its execution and returned an Object, Spring encodes it into JSON and sends the result back to the browser to be converted into a Javascript object graph. Note that all referenced objects (and their referenced objects, etc) are converted to JSON and included, via a configurable set of translation annotations.¹⁰ Be careful not to have any circular references, or Spring will loop back and forth attempting to encode the relationship and eventually fall into a StackOverflow situation. Once Spring sends the response back to the client, either the success() or error() method is activated based on the server result. If successful, the response.AJAXResult field will have been populated with the result of the controller method (and its related objects). If a failure, the error method will

¹⁰ Examples are @JsonProperty and @JsonAutoDetect; for more information,, refer to the Jackson API

	have been called and more information is available via the first JavaScript method parameter.
--	-----------------------------------------------------------------------------------------------

6.1.4. Reporting

PCMMR uses SQL Server Reporting Services (SSRS) to implement both canned and ad-hoc reports. When the menu option is clicked, SSRS browser interface is opened in a new browser window above PCMMR. The VA active directory credentials used in Windows and by CISS are intended to be reused transparently by SSRS, to display reports available to the user and restrict stations and other data elements for that user. SSRS can export reports in a variety of formats (PDF, Excel, CSV, etc).

6.2. Batch Processes

There are three types of batch processes which are managed by Spring Batch framework and execute operations over a large number of patients. Spring Batch provides a level of resiliency toward server crashes and exceptions, since it maintains a stateful context via its own database tables and thus can resume operations after being interrupted. For smaller sets of data, simple loops can be used in the code and a single transaction used since locking is less of an issue.

Spring Batch defines Jobs, each of which has multiple Tasklets which are executed in the order defined. PCMMR currently implements custom tasklets for each job, although Spring Batch supports simpler configurations. These jobs and tasklets are all configured in `pcmm-batchProcesses.xml`.

Cleanup Binary Objects Tasklet – Because Spring Batch operates over several container-managed iterations, it is not feasible to run the whole process within a single transaction. Additionally, Spring Batch supports primitive data types to be stored in a shared context across multiple iterations, but complex data types / Serializable data is not supported. Thus, PCMMR allows for a map of more complex data to be persisted in the database for the duration of the Spring Batch execution, and retrieved via a common key. The purpose of this tasklet is simply to delete any potential persistent data after the job completes

6.2.1. Transfer Patients to Entire Team

This batch process has five tasklet steps:

- **Perform Patient Transfer** – This is the tasklet which activates the shared `HandleBatchOperationsRequestTasklet` to perform the patient moves. Each patient is executed over a single iteration of the tasklet and a shared context is built during each iteration to maintain state and determine which patient to handle next. Each patient transfer (tasklet activation) makes a single transactional service call, so if one patient's execution fails, the error is captured but the overall process allowed to continue.
- **Unassign Staff Members** – Once the patients have all been transferred to the parent team, and only if all patient transfers executed correctly, staff members for the specified position are unassigned (both for the current time and any future assignments). As each staff member is modified, a background job log entry is created for reporting.

- **InactivateOrDeletePositionAfterPatientTransfer** – If the previous two tasklets completed successfully and all patients and staff were updated successfully, the position itself is inactivated or deleted based on the overall job request parameters. As above, background job log entries are created to detail what happened.
- **Create Patient Transfer Job Completion Alert** – Alerts are created in this tasklet which summarize the job execution results; if no errors occurred, only one alert is created, otherwise separate alerts are created based on the AlertTemplate rules defined in the database.
- **Cleanup Binary Objects Tasklet**– See description for this tasklet above.

6.2.2. Patient Bulk Operations

This batch process is similar to the one above but only includes three steps:

- **Perform Batch Operation** – Based on the parameters specified, this tasklet performs one of the following behaviors:

Process	Activated From
Bulk Assign – Assigns many patients to a team or position	<ul style="list-style-type: none"> • Historical Patient Assignment Listing • Patient Search Results
Bulk Unassign – Unassigns many patients from a team and/or position	<ul style="list-style-type: none"> • Patient Assignment Listing • Patient Position Assignment Listing
Bulk Team Move – Moves many patients across teams	<ul style="list-style-type: none"> • Patient Assignment Listing • Patient Position Assignment Listing
Bulk Position Move – Moves many patient assignments from one explicit position within a team to another explicit position in the same team	<ul style="list-style-type: none"> • Patient Assignment Listing • Patient Position Assignment Listing

- **Create Job Completion Alert** – As implied, this tasklet creates alerts based on the successful or unsuccessful results of the batch operation tasklet above.
- **Cleanup Binary Objects Tasklet** – See description for this tasklet above.

As in other batch operation jobs, a background job log is created detailing the actions for each patient modified and the overall success or failure of the job.

6.2.3. Cluster Considerations

PCMMR consists of multiple unattended servers, each of which has the capability of executing batch processes. A batch process execution is managed only by one unattended server until it completes. To activate any of these batch processes, a JMS message is sent to the Batch Process Queue (defined by the `jms.batchProcess.queueName` property in `pcmm.properties`), which activates the `handleMessage` method in the `BatchProcessListener` message-driven POJO. Because PCMMR uses distributed queues in a clustered environment, the application server container should evenly distribute requests for these batch processes across each unattended server's JMS queues, such that load is balanced. If an unattended server

crashes, it should be able to reactivate any partially-completed spring batch processes after it restarts (or potentially another unattended server can resume where the first left off).

6.3. Scheduled Jobs

PCMMR has many scheduled jobs defined which execute only on the unattended servers. Their schedules are configurable via `pcmm.properties` entries, either using cron triggers or fixed time delays between the stopping time of one execution and starting time of the next. These are all defined via the Spring `@Scheduled` annotation and by convention the scheduled job class names all end with `*ScheduledJob.java`.

Each unattended server executes every scheduled job using the same scheduling parameters (although their start times may be slightly staggered), so care was taken using HazelCast locks and shared data to ensure only one unattended server executes a job (or parts of a job) at a time.

6.3.1. MVIPatientInboundProcessorJob

This is a background job which periodically queries a HazelCast map containing MVI patient update requests, and processes them in order for each patient. Messages are retained for a configurable amount of time to ensure in-order processing, since the JMS transport mechanism doesn't strictly guarantee FIFO ordering. A separate MDB actually populates the HazelCast map when messages arrive in the JMS queues (after being transformed by Mirth Connect from the original HL7 format). Four types of messages are processed: ADTA24 ("patient unlink"), ADTA43 ("patient move"), ADTA31 ("patient primary view updates") and MFNM05 ("patient treating facility list update"). See section 7.2.2 for more details.

6.3.2. MVIPrimaryViewAndRegistrationScheduledJob

This is a background job which processes MVI registrations for any patient flagged in the PCMMR database as needing such. It processes each patient in a separate thread, and registers interest for that patient at all Vista sites where the patient is known to exist (via the treating facility list and existing objects in our VistaPatient database table). It waits for all threads to complete before exiting.

In order for this job to run for a given Vista station, the process at that site must be enabled via the flag stored in the `PCMM_VISTA_INSTANCE` table.

6.3.3. PurgeJobExecutionResultsJob

This is a simple background job which runs periodically and deletes any background job log entry older than a configurable amount of time (via `pcmm.properties`).

6.3.4. TeamValidationJob

This is a background job which performs data validations for teams flagged in the PCMMR database as needing such. It runs validations for each team in its own thread and stores the results in the database for later retrieval. All validators are executed, using the "team" scope (so that the entire Team object graph is validated, as opposed to the scope used when users perform actions on the front-end and we only are interested in validating their specific data update). Each team's validation occurs in a read-only transaction, which ensures a consistent view of the team as per when the transaction started, but without modifying the team so that users can continue to make data changes on the front-end UI without encountering versioning problems. The validation process as a whole, and also the subsequent Vista

synchronization process, are both designed to be “passive”, operating outside the normal JPA/Hibernate versioning process to avoid OptimisticLockingExceptions and database-level locks. They also take advantage of SQL Server’s “snapshot” isolation for read-only transactions – see <http://msdn.microsoft.com/en-us/library/tcbchxcb%28v=vs.110%29.aspx>.

Each team’s validation is executed in a separate dedicated thread. The threadpool is managed by Spring’s task executor abstraction, which is configured in the pcmm-scheduledTasks.xml file.

The complete process is outlined below:

```
batch process - runs on unattended servers - every 60 sec
  select team_id, team_version from teams
    where validation_needed = y order by record_modified_date
  lock hazelcast on "Validation" + team_id;
    if lock unavailable, assume other node is processing this team already
    and move on to next team
  begin DB transaction
  get full team obj where team_id and team_version match
    and validation_needed is Y
    if none, assume someone else made changes in the meantime; exit
      either a user, which would change the version, or
      another unattended node which ran the validation process and
      changed the validation_needed flag
  run validations on team
  if error
    rowsUpdated = "update team set error = X, validation_Needed = N
      where team_id = X and team_version = Y"
  if no error
    rowsUpdated = "update team set error = null, validation_Needed = N
      where team_id = X and team_version = Y"
  if rowsUpdated = 1
    send JMS message to vista_sync queue containing team_id and team_version
  if rowsUpdated = 0
    validations are outdated since version must have changed; it will be
    picked up by next process
  unlock hazelcast
```

6.3.5. VistaCleanupJob

The VistaCleanup job is intended to remove any data inadvertently stored in Vista fileman files when an error occurs during the Vista synchronization process. Since Vista is not a transactional resource, it’s possible that PCMMR could have synchronized only half of some Team’s object graph out to Vista before an error occurred on the PCMMR side that caused the process to fail. If the team is re-sync’ed later, the graph will be completed and corrected normally. However, if the team is first deleted on the PCMMR side, or the IEN of the newly-created Team object in Vista was never stored back in the PCMMR database for tracking, the object in Vista will become a “zombie”. The purpose of this job is to clean up such data elements.

This process was never fully built or enabled since no “team delete” function exists in PCMMR, and during testing the IEN for new Vista objects were reliably set in the PCMMR database regardless of errors occurring during synchronization. The skeleton code remains for this job in the event that it’s needed at some point. The job’s logic would roughly be to search for all Vista Team objects in the 404.51

file, and compare all IENs of those objects to all known Team IENs in the PCMMR database. Then it would delete any outliers.

6.3.6. PatientAutoInactivationJob

The patient auto-inactivation is a multi-step process which runs periodically to update patient assignments in PCMMR based on encounter data changes in Vista. It also retrieves events generated in Vista for when the patient date-of-death field is flagged, or rescinded. In order for this job to run for a given Vista station, the process at that site must be enabled via the flag stored in the PCMM_VISTA_INSTANCE table. A separate “last job execution date” column is also stored in this table, which is used to ensure updates don’t occur more frequently than expected due to the fact that PCMMR has multiple redundant unattended servers all concurrently running these processes.

After the process runs, the PCMMR database is updated with the last execution time of this job. The next time this batch process job executes, it will only query Vista for events which were created since the last execution time. This field in the database can be manually overridden with an earlier historical date if for some reason the job failed. It is safe to rerun this process for the same date range multiple times without concern for creating data inconsistency.

“Process Encounters” subprocess detail

When this process runs, it queries all encounters created in each Vista since the last time the process ran. If an encounter meets a set of criteria (based on the respective team’s care type, encounter stop codes, primary vs. secondary provider status, etc.), it serves to “flip” pending primary care encounters to “active” and update the first & last encounter dates for each patient assignment as necessary. It also triggers the same set of automatic events (multiPCP modifications, etc) used elsewhere in the system when the patient assignment goes active.

After processing encounters, the subprocess performs two other steps:

Flag Stale Primary Care Patient Assignments

This step searches for primary care patient assignments in PCMMR which have exceeded a time limit without a qualifying encounter being reported in Vista, and flags the assignment for auto-inactivation after another X days have elapsed. The purpose is to automatically remove patients from PACT teams where they are not seeing any teamlet member at least every N months. The values used are configurable via the pcmm.properties file. This action also triggers alerts when a patient is flagged.

Inactivate Expired Primary Care Patient Assignments

This step searches for primary care patient assignments in PCMMR whose “dates flagged for inactivation” have expired – that is, they are on or before the current date. It unassigns the patient from the team and creates alerts as required.

As the patient auto-inactivation process runs, a new background job log is created showing each individual action that occurred. These jobs are available via the View Background Job Log menu item for the configurable amount of time they are retained on the system.

The set of data updates for any given Vista station are executed in a separate dedicated thread. The threadpool is managed by Spring's task executor abstraction, which is configured in the pcmm-scheduledTasks.xml file.

6.3.7. CPRS Header Sync Job

This background job calculates and writes new primary care assignment information for each patient to his/her respective "Outpatient Profile" (OP) entry in the Vista sites where he/she is known. The value stored in the OP is retrieved by CPRS and displayed directly in line 1 of the header box of the CPRS patient cover page. In order for a patient's data to be updated, the CPRS header sync process at that site must be enabled via the flag stored in the PCMM_VISTA_INSTANCE table. A separate "last job execution date" column is also stored in this table, which is used to ensure updates don't occur more frequently than expected due to the fact that PCMMR has multiple redundant unattended servers all concurrently running these processes.

When data changes are made in PCMMR, VistaPatients are flagged as needing new CPRS header information calculated. The lines for each patient may or may not change; if it is recalculated but the new value isn't different than the old value, no Vista operation is performed. The latest cached value for each Patient for a Vista site is stored in the VistaPatient PCMMR database table.

Updates to all VistaPatients for any given Patient are executed in a separate dedicated thread. The threadpool is managed by Spring's task executor abstraction, which is configured in the pcmm-scheduledTasks.xml file.

Pseudocode for the CPRS header sync job logic:

```
request update logic:
    set cprs_headerRefreshNeeded = true, version = version + 1

batch process logic- every N minutes:

open read-only transaction
get all {patient, VistaPatient + version} tuples needing a refresh
    (skip any vistaPatients whose cprsHeaderLastSynchronizedDate is > (N-1) minutes
    ago - some other node recently updated it)
    (skip any vistaPatients whose CPRS header sync process for their respective
    station is turned off)
group by patient
for each patient:
    attempt lock on patient ID (to prevent concurrent update on another unattended
    node)
        if lock fails, continue to next patient

        for each VistaPatient in Patient
            calculate new header text
            if different than current header text
                add {VistaPatient, new header text} to var "updatesMap"
        unlock patient ID
close transaction

for all the updates in updatesMap
start async:
    lock on patient assignment ID
    open read/write transaction
    update header text only if version still matches
```

```

        if rows updated = 1
            write header out to Vista and commit transaction
        else rollback transaction (version was incremented; someone else updated it,
wait for next batch)
        release lock

```

6.3.8.CPRS Header Sync Monitor Job

CPRS shows the PACT assignment information for a patient in line 1 of the CPRS header box on the patient cover page. This information includes the name of the provider and associate provider. The provider (staff) information may change automatically in PCMMR via a background scheduled job; there is no user-driven event that we can use to capture when a staff member's name changes. Therefore PCMMR uses this Header Sync Monitor Job to query for any changes to staff member information in the PCMMR database and request a new CPRS header synchronization for all patients showing that staff member information in their current header.

The parameter used to signify when this process was last executed is stored in table called PCMM_JOB_EXECUTION_METADATA.

6.3.9.Staff Update Job

The staff update job runs periodically to update staff information in the PCMMR database with the staff information (on the "new person" file 200) in each Vista station. In order for staff data to be updated, the Staff Update process at that site must be enabled via the flag stored in the PCMM_VISTA_INSTANCE table. A separate "last job execution date" column is also stored in this table, which is used to ensure updates don't occur more frequently than expected due to the fact that PCMMR has multiple redundant unattended servers all concurrently running these processes.

A separate dedicated thread is allocated for each Vista station's data updates. The threadpool is managed by Spring's task executor abstraction, which is configured in the pcmm-scheduledTasks.xml file.

The logic for this job is to query for ALL staff members at a site, and iterate over them to see if they exist in the PCMMR database. If they do, compare all fields to determine if any updates need to be made and update the DB if needed. After information for a staff member has been updated, a background job log entry is created.

Additionally, if the staff member's person class changed or expired since the last time they were updated, PCMMR sends out alerts to the PCMM Coordinator requesting they manually unassign the staff member from their respective teams (as displayed in a table on the Staff member profile page). If the staff member's termination date is set, alerts are also sent to PCMM coordinators and the staff member becomes unavailable for assignment on the PCMM UI. They are not automatically unassigned from teams/positions.

6.4. JMS / MDB's

PCMM uses JMS messaging provided by the container (WebLogic) to ensure reliable message delivery for some processes. Inbound message processing for message-driven POJO's is configured using Spring's namespace-driven "listener-container" (see <http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#jms-mdp>) and the typical JmsTemplate pattern for outbound message

production (see <http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#jms-jmstemplate>).

6.4.1. Batch Process Events

This is a generic listener which activates the Spring Batch process given a JMS message. The message contains parameters specifying which job to run, what object IDs to process in the job and other “plumbing” fields like the requesting user ID and Vista authentication token to use. Binary objects sent using a dedicated JMS message key are automatically persisted in the DB for later reference in the Spring Batch job if needed.

6.4.2. Vista Update (Synchronization) Event

This is a simple JMS sender/receiver which manages requests to synchronize a PCMMR Team object graph with Vista. It captures the team needing synchronization and delegates the incoming request to the VistaSynchronizationServiceImpl class. JMS was used as a wrapper for this process so that requests could be distributed easily across multiple unattended servers, and so that a certain level of resiliency was met if the system went offline. The JMS store-and-forward queue configuration ensures the sync requests aren’t lost. See section 6.5 for more details on this process.

6.4.3. MVI Service

This is a simple JMS receiver which manages requests to update patient information given incoming MVI messages. It is a thin wrapper / delegate which accepts the incoming message and immediately stores it in a HazelCast map for later consumption by the MVIPatientInboundProcessorJob. See section 6.3.1 for more details on this process.

6.5. Vista Synchronization

PCMMR is the authoritative source for all patients and the Care Teams to which they are assigned. PCMMR implements a national view of patient care teams and provides a critical function for patient management and care, allowing the display of each patient’s Care Teams (regardless of which Vista Site they are assigned to) in the CPRS header and the CPRS Primary Care window. In addition to CPRS, there are many other Vista applications that rely on data sourced in the PCMM Vista files, including MyHealtheVet, DSS and Scheduling/Appointment Management. For this reason, PCMMR synchronizes a majority of its enterprise data back to the individual Vista systems and stores the data in the same PCMM files used by the legacy application. It does this via VistaLink, invoking mostly PCMM and Scheduling-namespaced RPCs. The synchronization of this data prevents other applications from needing to request data from the new PCMMR data store. It also alleviates a performance bottleneck since CPRS is accessed nationally at such a high rate that PCMMR wouldn’t be able to keep up with its incoming requests for team information.

The Vista synchronization process occurs after the Team Validation job runs (see section 6.3.4). Teams are only synchronized to Vista if they don’t have validation problems, to ensure data consistency. In order for this process to run for a given Vista station, the process at that site must be enabled via the flag stored in the PCMM_VISTA_INSTANCE table. A “last vista synchronization date” field is set by this process for each team (in the TEAM database table), and is reported in the Team profile screen at the bottom. Also a background job log entry is created, with individual entries describing what elements were created/updated/deleted in Vista during the synchronization process, and their respective IENs.

The synchronization process as a whole is intended to operate passively, outside the normal JPA/Hibernate versioning system, so that it doesn't interfere with users making concurrent changes to the team on the front-end. As users make those UI changes, the team object in the database is re-flagged for validation / synchronization, and the version incremented. When the background job is finished synchronizing one version of the team, it only sets the "sync needed" flag back to "N" if the version still matches what it was at the beginning of the synchronization process; otherwise, some other process or user must have since modified the team and another synchronization is needed. However, if a data element is written to Vista and Vista assigns it an IEN, that IEN is stored back to the PCMMR database regardless of the version of those objects, to ensure Vista data entries aren't "lost". The IEN fields stored in the database are considered scoped to the Vista Sync process itself, and aren't updated via the front-end UI changes, so no conflicts should occur.

The complete process is outlined below:

```
JMS vista sync queue listener: incoming team_id, team_version
    lock hazelcast on "VistaSync" + team_id
    create read-only DB transaction
    get team with same ID and version, prefetch children
        if none exist, exit, we'll sync after the next update
    ensure vistasync active for team's station
    push changes to Vista
        assign IENs in new transactions without incrementing team version or
        caring what our DB versions are
    update last vistasync date without version increment on team
    unlock hazelcast
```

6.6. Performance Enhancements

6.6.1. Hibernate level 2 cache – ehcache

The Hibernate level 2 cache is described at <https://docs.jboss.org/hibernate/orm/4.0/manual/en-US/html/performance.html#performance-cache> and is implemented in PCMMR via EHCACHE. It is configured in the file ehcache-pcmm.xml. In this file are various caches for objects and their children collections.

6.6.2. JPA batch fetch

JPA batch fetch is a way to prefetch many children of an object using a separate dedicated query for retrieving N children at once, as opposed to running a separate query to retrieve each individual child as it's accessed. It is described in detail at http://en.wikibooks.org/wiki/Java_Persistence/Relationships#Batch_Fetching. An example of how this is configured in a PCMM model object is at the accessor method level:

```
@OneToMany(mappedBy = "team", fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
@BatchSize(size = 40)
@JsonProperty
public List<Position> getPositions() { ...
```

As shown, the @BatchSize annotation on the getter for the child collection signifies that Hibernate should run a query to retrieve Position children in sets of 40.

6.6.3.JPA join fetch & QueryCustomization

Join fetching is described at http://en.wikibooks.org/wiki/Java_Persistence/Relationships#Join_Fetching and is a way to proactively select data for children of an object in the same query that's used to select data for the parent. PCMM implemented an abstraction for this type of optimization in the QueryCustomization class, which encapsulates a set of customizations for running a query. For those given DAO methods that support a Java 7 varargs QueryCustomization parameter, the caller can specify which children should be proactively fetched for that call. No other duplication or customization of the DAO Java method is necessary.

An example looks like:

```
List<PatientAssignment> pcAssignments = patientAssignmentDAO
    .findAssignmentsForCriteria(null, null, null, null,
        null, patient.getId(), null, PRIMARY_CARE,
        null, null, Arrays.asList(ACTIVE, INACTIVE),
        null, new QueryCustomization(TEAM));
```

This signifies to the DAO that the caller is retrieving a set of PatientAssignment objects, but in each object the Team field should be join-fetched and pre-populated. The TEAM enum value passed to the QueryCustomization constructor is an instance of a class whose name ends with “AssociationFieldType” (by PCMM convention) and resides in the gov.va.med.pcmm.persistence.queryCustomization package. These enums simply represent the field names for associations or children collections contained in the main PCMM model objects.

6.6.4.DB indexes

Database indexes were added to support specific long-running queries after performance testing was implemented and results analyzed. One example is an index could be {IEN, station number} for those tables which contain these columns; they are often criteria used to search for items used during Vista synchronization and/or MVI inbound message handling.

6.6.5.Multiple unattended servers

PCMM has many background scheduled jobs and batch processes, in sections 6.2 and 6.3. These processes operate over 130+ Vista stations, which is the reason that multiple unattended servers are used to run them concurrently. Care was taken to ensure each unattended server operates independently on any given item, typically using HazelCast locking – see the gov.va.med.pcmm.service.cluster.ClusterLockType enum values for uses of this type of locking.

Any unattended server can be taken offline, or more servers added to the mix, to support horizontal scaling of the running application. HazelCast can transparently adopt new servers at runtime and WebLogic will distribute JMS messages per its cluster configuration accordingly.

6.6.6.Spring @Async / TaskExecutors

Many processes in PCMM operate over large data sets and for performance reasons need to run fragments of code in separate concurrent threads. Spring has a clean design for this requirement which uses a combination of its @Async method annotation and task executors, which delegate to internal configurable thread pools for efficiency. See <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/scheduling.html> – PCMM configures these items in the pcmm-scheduledTasks.xml file.

7. External Interface Design

PCMMR communicates with several other systems, such as individual VistA installations (via the VistaLink tool), MVI via web services and HL7, LDAP for general user authentication and SMTP for sending emails. A summary of all interface systems is contained in the following embedded spreadsheet (and is subject to change):



PCMM Interfaces

7.1. Interface Architecture

- **VistA** - Across the VA network, over 100 separate installations of VistA exist, inside VA Medical Centers (VAMCs), Community-Based Outpatient Clinics (CBOCs), and elsewhere. PCMMR communicates with these different sites over a socket connect via the VistaLink tool, which is a shared WebLogic resource adapter.
- **MVI** – MVI is the authoritative source for patient identity trait information, such as name, social security number and date of birth. It correlates and publishes changes to this identity information to various systems within the VA, including all VistA sites. PCMMR interfaces with MVI to ensure that the local data stored by PCMMR is in sync with the “primary view” - the master authoritative copy - such that when users perform searches against the data stored in the PCMMR enterprise database, they can be assured they will match the official latest version from MVI.

In order to perform this synchronization, PCMMR subscribes to HL7-based messages published by MVI. These update messages can be both primary view updates as well as split and merge requests for cases when two or more patients are mistakenly linked to a single record, or when two or more records are linked to a single patient. See section 7.2 for details.

- **CPRS** – PCMMR writes to the Outpatient Profile file for each patient at each Vista site, which is then read by CPRS when displaying PACT information in line 1 of the header box on the CPRS patient cover page.

When the user clicks the header box in CPRS, it calls a PCMMR webservice which returns a complete set of information for the patient in XML format. A Vista RPC then transforms this XML into the content shown in the popup window that appears.

- **LDAP** – At the CISS level, when any user logs in, the system authenticates the user credentials and assigns a set of roles and permissions for later authorization based on the data stored in the VA enterprise LDAP server. The roles and permissions within LDAP are synchronized with the roles assigned in the PCMMR enterprise database.
- **SMTP** – Notification functionality in PCMMR (and indeed across all CISS partner applications) relies on an SMTP server for sending email. The main VA SMTP server is used for this purpose. During production deployment, a separate linux SMTP proxy may be used to achieve “retry” logic in the event that the main enterprise SMTP server is down for maintenance or can’t be reached on the network.

7.2. Interface Detailed Design

7.2.1.Vista

Deployed separately from any application, VistaLink is shared by all enterprise Java applications in WebLogic and provides to them an abstraction on top of the socket connection, such that they can run remote procedure calls (RPCs) and send & receive data to/from each individual Vista instance.

In PCMMR, the VistaController class contains methods that use VistaLink to run various RPCs, which in turn relies on the RpcServiceUtilFactory class for utility methods. The approved PCMMR application proxy user is used for all data interface requests to Vista; auditing information is stored and managed in PCMMR's database history tables (those ending in _H).

PCMMR has the ability to search the patient and staff population at the specific Vista site level (in addition to searching its own local database). It also can query for patient auto-inactivation events, recent patient encounters and events logged when the date-of-death field is set or unset.

7.2.2.MVI

The integration pattern used by the PCMMR software solution is the "Decentralized Hybrid pattern", defined in the MVI Service Description Document. This pattern allows PCMMR to interact with MVI to retrieve data.

Messages are sent to MVI from PCMMR and vice-versa using the HL7 format. The older style (v2.4) and newer style (v3) of this format are both used, and they are transported along different protocols (HTTPS/SOAP web services, MLLP) based on the capabilities of the MVI services at the time of integration. Mirth Connect integration engine transforms inbound and outbound HL7 messages and converts them to JMS messages on the fly for processing by PCMMR. For details and example request bodies, refer to the latest MVI Service Description Document.

A business requirements flow was co-developed by the PCMMR and IAM teams during the creation of the iRSD, which can be seen in the appendix section 10.3.

PCMMR will have a dedicated station number assigned to it with a 200-series prefix. Based on the different events described here and in the iRSD, PCMMR will be communicating with MVI as per the following table:

Call / Event	Direction	Format	Message Code / Web Service call
get Primary View + get Corresponding IDs composite call	PCMMR -> MVI	HL7 v3	1305 MVI Comp1
Register PCMMR as system of interest for patient	PCMMR -> MVI	HL7 v3	1305 MVI Comp1 with "Register Interest" flag passed
Patient identity information updates	MVI -> PCMMR	HL7 v2.4	ADT-A31
Treating facility updates	MVI -> PCMMR	HL7 v2.4	MFN-M05

ICN merge resolution- more than one person has the same ICN and a split has been done	MVI -> PCMMR	HL7 v2.4	ADT-A43
ICN duplicate resolution – one person has more than one ICN	MVI -> PCMMR	HL7 v2.4	ADT-A24

When communicating with MVI, generally the fully-qualified VistA source ID is used. This consists of a concatenation of several tokens, including the DFN (also known as the Internal Entry Number or IEN) and station number.

To query MVI, an HL7 request is created and sent to the appropriate endpoint in the MVI system. The VAAFI security layer is utilized for all environments except development. It encrypts both data sent to and received from the MVI server. For details of the HL7 message format, protocol, and example request bodies, refer to the latest MVI Service Description Document. For additional requirements and the workflow of the “Manage a Patient Panel Request” use case, see section 10.3.

Several use-cases have been defined in the PCMM-MVI Integration Requirements Specifications Document (RSD):

- a. *Managing a Patient Panel Request* - In order to receive current identity data from MVI, PCMMR will use a VistA Source ID to send MVI a Retrieve Person with Corresponding IDs request for each initial Patient Panel Request.

PCMMR shall pass the “Register Interest Flag” in the Retrieve Person with Corresponding IDs request. This will enable PCMMR to receive identity updates from MVI.

MVI will return a copy of the person’s Primary View (a collection of identity data elements, including the ICN) and provide updates to that data as they occur in MVI. MVI also will provide a list of identifiers associated with that person in MVI. This will allow PCMMR to determine whether a patient in PCMMR is known at other VA Medical Center (VAMC) sites.

- b. *Update Person Identity Traits* – The Update Person Identity Traits business activity involves the PCMMR receiving and processing updates from the MVI service to maintain up-to-date person identity data.

Upon receipt of a message, PCMMR will activate an internal service call which searches for the affected patient (via VistA source ID) and updates the patient’s identity data within the PCMMR national database. After successfully updating this data, PCMMR will send an acknowledgement HL7 message back to MVI signaling that the initial message was processed.

- c. *Process “Resolve Duplicate ICN” Maintenance Message* - The Resolve Duplicate ICN business event involves PCMMR receiving and processing link requests from the MVI service in order to resolve situations where one person has multiple ICNs.

Upon receipt of a message, PCMMR will activate an internal service call which searches for the separate patients and resolves the duplicate data within the PCMMR national database. The resolution steps include, but are not limited to:

- Moving one patient's team assignments to the other patient
- Deleting the bad patient from the database (the old records will permanently exist in the "_h" database history tables for auditing purposes)
- Updating the patient's identity trait information with the new authoritative (merged) data stored in the MVI primary view
- Sending an alert to the PCMMR coordinators at the affected sites notifying them of the patient merge and subsequent team changes (as well as any business rules now violated, such as duplicate PACT assignments)

After successful resolution of the duplicate ICN situation, PCMMR will send an acknowledgement HL7 message back to MVI signaling that the initial message was processed.

A duplicate ICN message can originate from either the MVI enterprise system or from any of the Vista stations. Due to the fact that PCMMR synchronizes data back to the individual Vista systems, it is critical that PCMMR does not merge patient data before the Vista system itself does; doing so would risk corrupting patient assignments and other PCMMR data elements. Therefore, upon receipt of a “Resolve Duplicate ICN” message, PCMMR performs a preliminary lookup into the appropriate Vista system to ensure the merge already occurred. For more workflow details, please refer to the embedded “MVI Detailed Design” document at the end of this section.

- d. *Process “Resolve ICN Mismatch” Maintenance Message* – The Resolve ICN Mismatch business event involves the PCMMR receiving and processing mismatch broadcasts from the MVI service to resolve situations where more than one person is linked to the same ICN.

Upon receipt of a message, PCMMR will activate an internal service call which searches for the affected patient (via ICN) and splits the single record into the appropriate multiple patient records within the PCMMR national database. The resolution steps include, but are not limited to:

- Creating a new patient record
- Optionally moving one or more team assignments to the new patient record, if business rules allow
- Updating both patients' identity traits with the new authoritative data located in the MVI primary view
- Sending an alert to the PCMMR coordinators at the affected sites notifying them of the patient split and subsequent team changes (as well as any business rules now violated, such as each patient having a required PACT assignment)

After successful resolution of the ICN mismatch, PCMMR will send an acknowledgement HL7 message back to MVI signaling that the initial message was processed.

Detailed information about the specific messages and application logic can be found in the following embedded document:



PCMMR_MVI_Detail
ed_Design.docx

7.2.3. CPRS

When the patient header box is clicked, CPRS uses the MyHealtheVet WebService Client (MWSC) Vista tool to perform an HTTP GET request to PCMMR. The request URL is in the format:

[Redacted URL]

eg.

[Redacted URL]

An example of the response returned looks like:

```
<PatientSummary>
  <PatientNationalAssignments>
    <PatientStationLevelAssignment>
      <StationNameAndNumber>CHEYENNE VAMC (#442)</StationNameAndNumber>
      <PrimaryCareAssignments />
      <MentalHealthAssignments />
      <OEFOIFAssignments />
      <SpecialtyAssignments>
        <SpecialtyAssignment>
          <CareTypeCode>4</CareTypeCode>
          <CareTypeName>MENTAL HEALTH</CareTypeName>
          <teamName>cmg *MH* CH12</teamName>
          <TeamMembers>
            <teamRoleName>(MHTC) PSYCHIATRIST</teamRoleName>
            <name>Hollar,Saul K</name>
            <phone>(999) 999-9999</phone>
            <pager>(666) 666-6666</pager>
          </TeamMembers>
        </SpecialtyAssignment>
        <SpecialtyAssignment>
          <CareTypeCode>4</CareTypeCode>
          <CareTypeName>MENTAL HEALTH</CareTypeName>
          <teamName>cmg *MH* Inpatient 442</teamName>
          <TeamMembers>
            <teamRoleName>ADDICTION THERAPIST</teamRoleName>
            <name>Thornhill,Carmela W</name>
            <phone xsi:nil="true" />
            <pager xsi:nil="true" />
          </TeamMembers>
          <TeamMembers>
            <teamRoleName>NURSE PRACTITIONER</teamRoleName>
            <name>Hubertus,Lourdes L</name>
            <phone xsi:nil="true" />
            <pager xsi:nil="true" />
          </TeamMembers>
        </SpecialtyAssignment>
      </SpecialtyAssignments>
    </PatientStationLevelAssignment>
  </PatientNationalAssignments>
</PatientSummary>
```

```

        </SpecialtyAssignment>
      </SpecialtyAssignments>
    </PatientStationLevelAssignment>
  </PatientNationalAssignments>
  <NonVAProviders />
</PatientSummary>

```

PCMMR uses a standard Spring controller along with the Jackson marshaller to serialize an object graph for this patient into XML and send it back to the Vista caller. All controller and model classes supporting this web service are stored underneath the gov.va.med.pcmm.web.wsendpoints Java package.

8. Human-Machine Interface

The user interface is via web browser. Internet Explorer and Firefox have been tested and verified, although IE 8-11 is the approved VA standard). The application user interface will be section 508-compliant.

8.1. Interface Design Rules

Basic common web UI rules were considered when designing the user interface for this application. The use of common HTML inputs such as radio buttons, checkboxes and text input boxes maps to the type of data being entered, such as a single selection, multiple selection, free text, etc. A common look-and-feel is shared across all screens, such as a unique title for each page, a main menu and set of breadcrumbs across the top title bar of the application, a footer row containing contact information and CISS links which allow the user to log in and out and switch to other portlet partner applications such as OHRS.

An analysis was performed by a Human Factors Engineering (HFE) team, and the results sent back to the PCMMR team in the following document:



The PCMMR team strives to make most or all of these design improvement suggestions by the time the final product is delivered to the customer.

8.2. Inputs

Inputs made to the PCMMR application are performed inside the web browser via HTML forms and submitted either with a common form submission button or via other image buttons using AJAX functionality.

If the data submitted by the user is incorrect or fails application validation, an error message is displayed to the user next to the field. For example:

CISS Portal Desktop - Mozilla Firefox

File Edit View History Bookmarks Tools Help

CISS Portal Desktop

AFHL CISS Local PCMM HP Sites VA Sites EEM Connor's HP Room WebLogic 10.3.2 docs

PCMM Patient-Centered Management Module

Station: CHEYENNE VAMC (#442) User Name: [REDACTED] Log Out

Parent: VISN 19

This is a TEST environment. Do NOT use or enter real Patient Data.

CISS Home OHRS PCMM

Home Patient Management Team Management Staff Reports Alerts Help Reference

PCMM Home Team List Edit Team Position List Edit Position

Modify an Existing Team Position

Please fix the errors below:

Team Profile:

Name: Alpha10 Care Type: PRIMARY CARE Focus: Status: Active Assignments: Open

General

Team Role: * CHAPLAIN

Description: PCMM Smoke Test

Expected FTEE: * 19

Expected FTEE: * Expected FTEE can not be greater than 1.00

Team Placement

☒ Team Lead

☒ Primary Team Contact

☐ Secondary Team Contact

Preceptor

Staff Role: (no preceptor)

Staff Name:

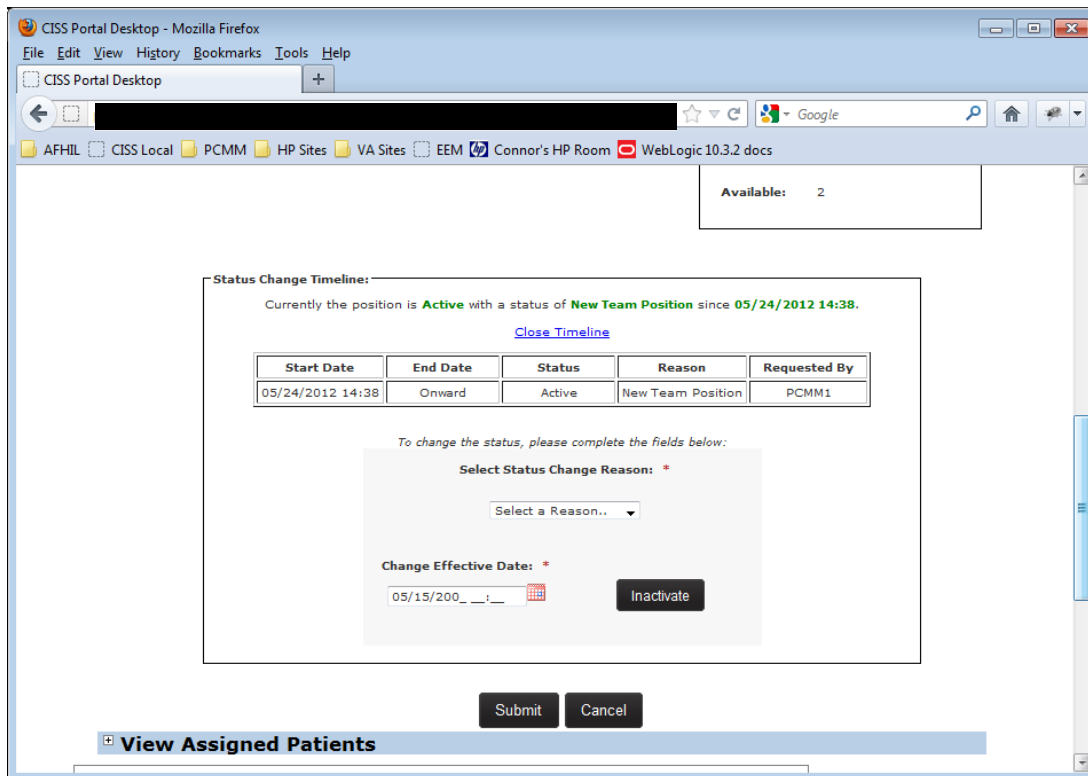
Patient Capacity

Allowed: * 2

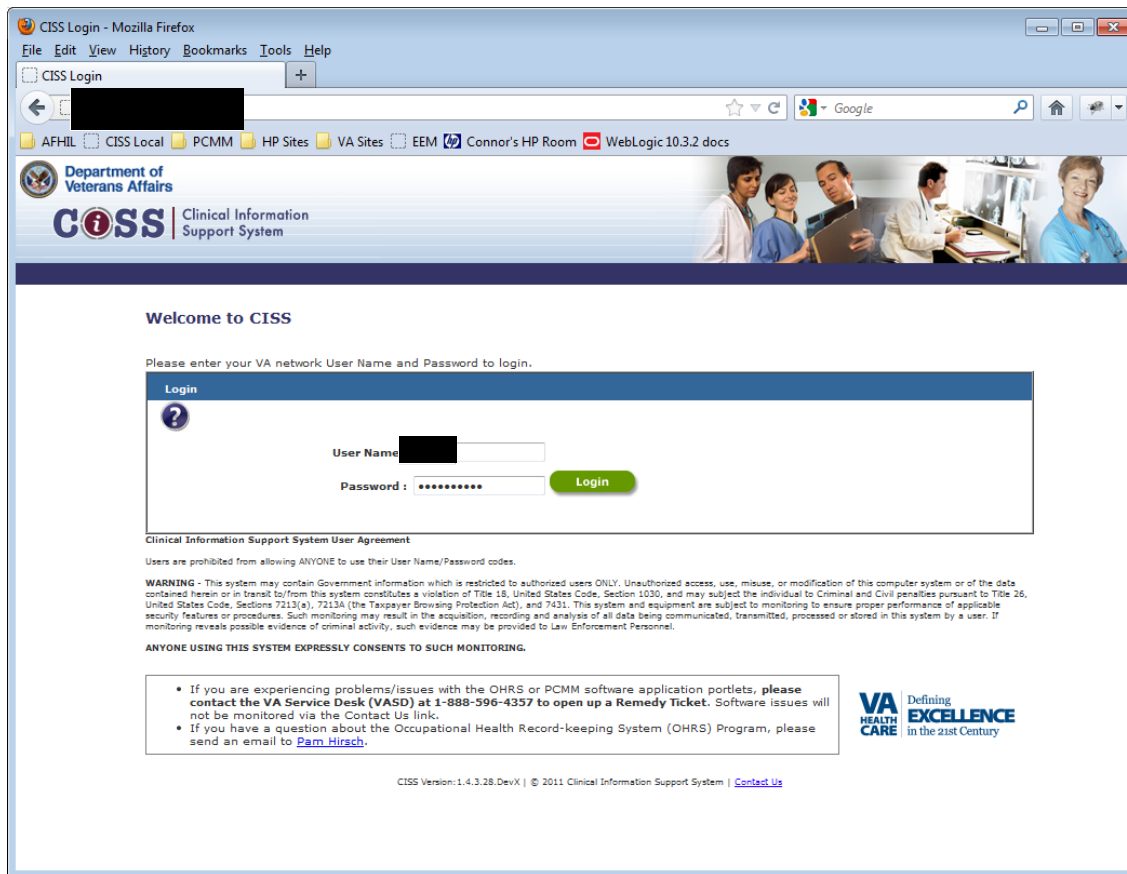
Assigned: 0

Available: 2

Some validations are performed on form submission but before the page input values are actually sent to the server (a performance enhancement). Additionally, some fields have masks defined which guarantee that the data is entered in the correct format. For example, a date field shows underscores for the different date/time values as the user is entering them, so the correct format is adhered to:



The only secured input field in the application is the password field on the CISS login screen, which follows the common HTML standard of masking the data entered as a set of bullets:

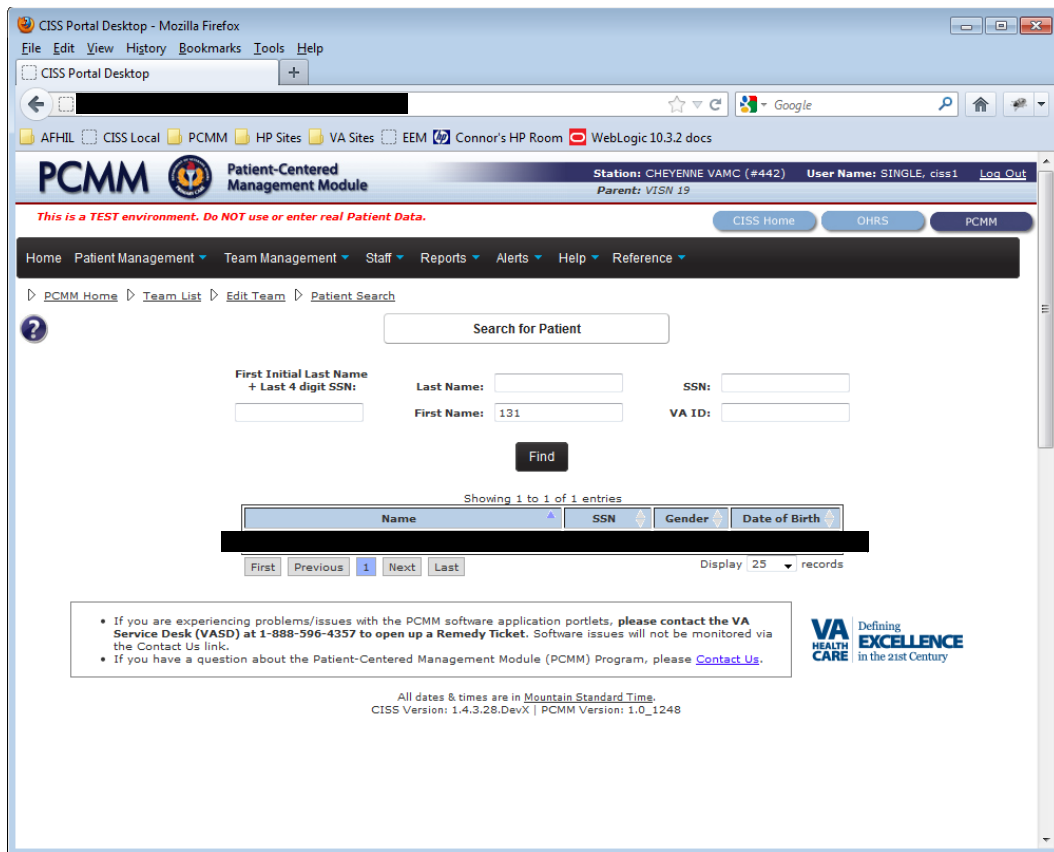


8.3. Outputs

Output of data in PCMMR is performed mainly in the web browser window, and generally is presented in two formats:

- Search results and lists of data are displayed in a standardized table format, which has embedded controls for sorting on any column, filtering on sensible columns via dropdowns at the eheader or footer level, and paging via a set of controls which group results into groups of 10-30 items. For example:

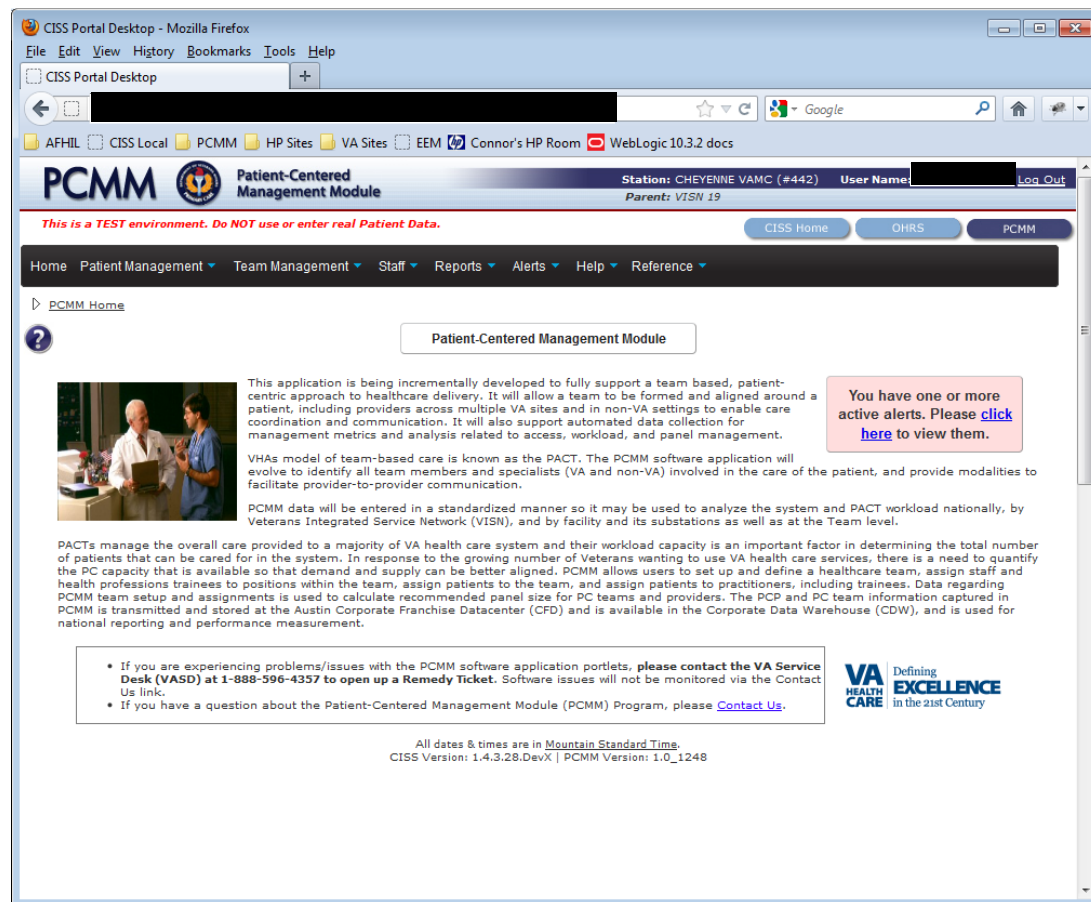
For security considerations, the social security numbers are sometimes masked to only show the last four digits. This only serves to add a thin layer of obfuscation for casual access to these screens since the privacy standard is on a need-to-know basis. To see the full number, or to search on sensitive fields in query screens, the user can switch to a detailed profile view of the data.



Reports are generated by SqlServer Reporting Services (SSRS) in a variety of formats. These reports are in a tabular format and can contain both detailed data (patient-level) and summary data (aggregate counts or sums across a variety of categories – teams, duty stations, etc.).

8.4. Navigation Hierarchy

After logging into the CISS portal and activating the PCMMR portlet partner application by clicking on the “PCMM” button on the top header bar, the user is presented with the PCMMR homepage:



As discussed in the previous section, common interface components are shown on this screen, such as:

- The main menu bar across the top, having a black background
- The breadcrumbs directly underneath – immediately after login, the only value is “PCMM Home”
- The Online Help activation icon – blue question mark directly below breadcrumbs
- The page title, shown centered and surrounded with a rounded rectangle
- The main page content
- The footer bar, consisting of support contact information, the VA logo, and the currently detected user time zone and application versions

Navigating throughout PCMMR involves selecting an item from the top menubar or returning to a previous screen by clicking on one of the breadcrumbs.

8.4.1. “Profile” example screen

The following shows an example of viewing detailed profile information for an object in PCMMR. This particular screen shows Team details:

CISS Portal Desktop - Mozilla Firefox

File Edit View History Bookmarks Tools Help

CISS Portal Desktop

AFHIL CISS Local PCMM HP Sites VA Sites EEM Connor's HP Room WebLogic 10.3.2 docs

Modify an Existing Team

[View Positions & Staff Assignments](#)

Team: * Alpha10

Station: * CHEYENNE VAMC (#442)

Care Type: * PRIMARY CARE

Focus: Please select...

Description: PCMM Smoke Test

Point of Contact - Administrative

Name: *

Phone Number: *

E-mail:

Point of Contact

Name: *

Phone Number: *

E-mail:

Patient Capacity

Assignment Status: * ☒ Open ☐ Closed

Modeled: * 1200

Assigned: 0

Available: 1200

☐ Allow Override

Adjusted:

Justification:

Reconcile the team against applicable models? ☒ No ☐ Yes

Status Change Timeline:

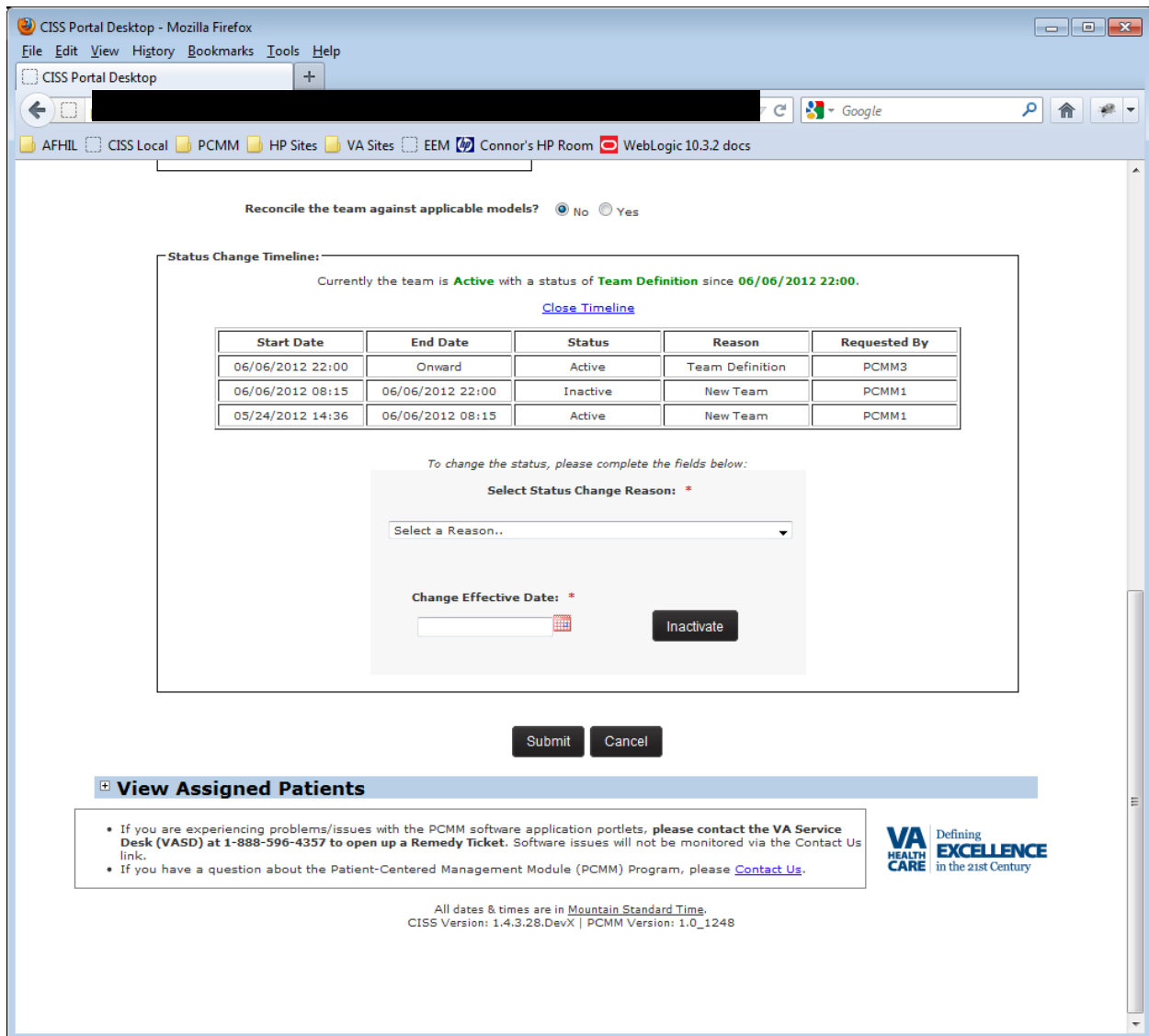
Currently the team is **Active** with a status of **Team Definition** since **06/06/2012 22:00**.

[View/Edit Complete Timeline](#)

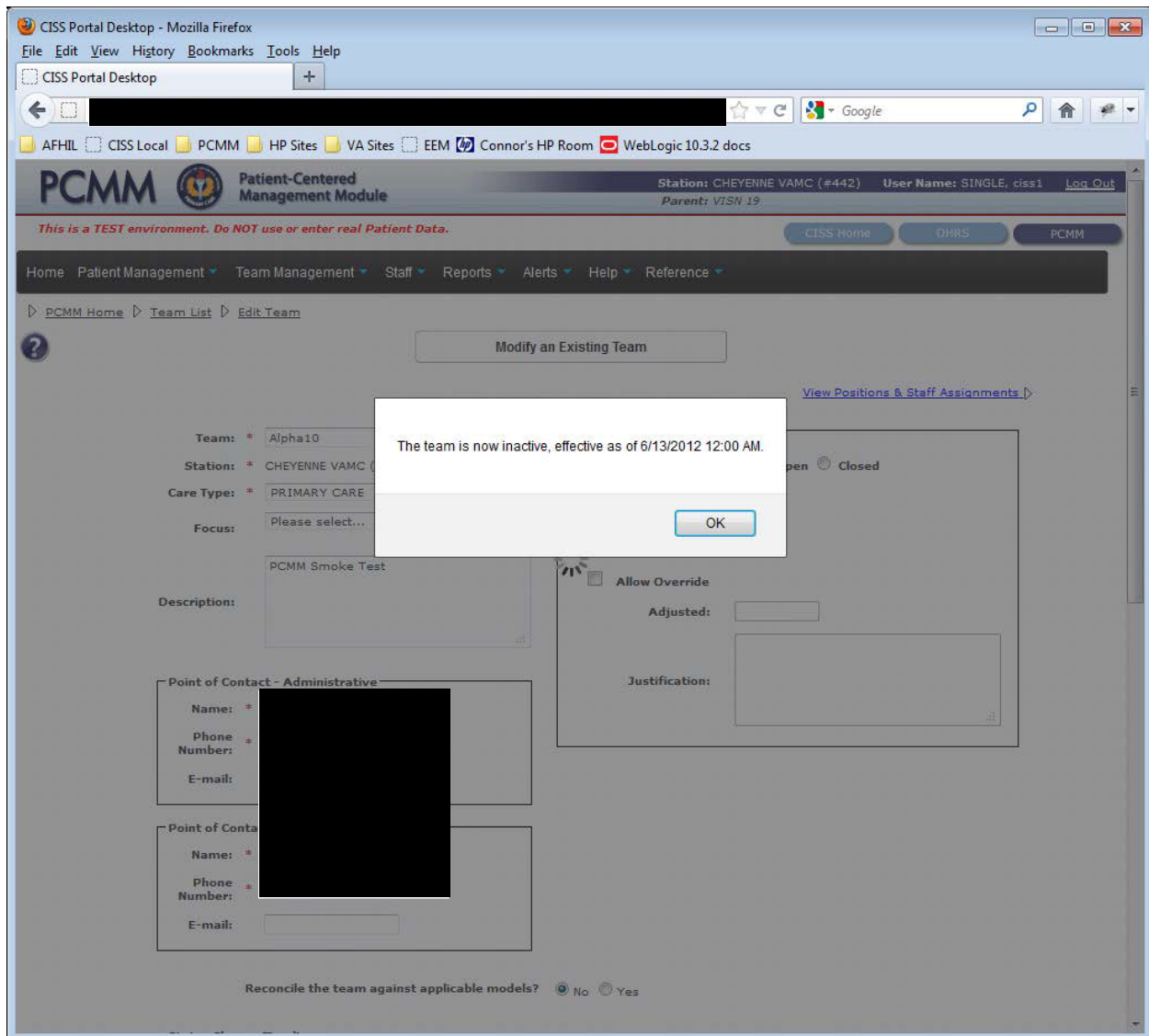
Submit Cancel

As displayed, fields are grouped into boxes of field sets, and capture details for the particular object. Because the user has permissions to update this object, the values are shown inside HTML input fields, which can be updated and the changes saved by clicking the “Submit” button at the bottom. Had the user not had edit ability for this team, all values would be simply displayed as fixed text and the Submit button would not be present.

The Timeline field shown at the bottom is initially hidden, but can be shown by clicking the “View/Edit Complete Timeline” link. When open, it looks like:



The table shown displays the complete history for this object, and the embedded controls (in the gray background box) allow new entries to be added to the timeline/history. The history table is sorted in descending chronological order. When a change is added to the timeline, it's done via AJAX such that the page doesn't need to be completely submitted. After the change is saved, the table automatically updates in the page and the user is presented a confirmation dialog box:



8.4.2. “List” example screen

Many screens in PCMMR also provide the user the ability to search for objects using a variety of criteria. The search results are displayed in a table format, which is consistent throughout the application. The search screen itself can either be a normal webpage or a popup dialog box. An example of the latter is the Team Search popup dialog box, which looks like:

CISS Portal Desktop - Mozilla Firefox

File Edit View History Bookmarks Tools Help

CISS Portal Desktop

AFHIL CISS Local PCMM HP Sites VA Sites EEM Connor's HP Room WebLogic 10.3.2 docs

PCMM Patient-Centered Management Module

Station: CHEYENNE VAMC (#442) User Name: SINGLE, ciss1 Log Out

Parent: VISION 19

This is a TEST environment. Do NOT use or enter real Patient Data.

CISS Home OHRS PCMM

Search for Team

Name: t

Search

Showing 1 to 25 of 30 entries

Name	Primary Care Provider	Care Type	Focus	Patients Allowed	Patients Assigned	Patients Available	Select
CB-Test1		SUB SPECIALTY (MED/SURG)	Dental	1200	0	1200	
CG *PACT* 442		PRIMARY CARE	Primary Care Only	1202	0	1202	
CG *PACT* WH 442		PRIMARY CARE	Womens Health	1250	0	1250	
MadTest		MENTAL HEALTH TREATMENT	Mental Health	1200	0	1200	
MM *MD* Team		COMMUNITY CARE	State Veterans Nursing Home	1200	0	1200	
MM *NC* Team		CASE MANAGEMENT		1200	1	1199	
MM *NJ* Team		PRIMARY CARE	HBPC	1200	0	1200	
MM *NY* Team		CASE MANAGEMENT		1200	0	1200	
MM *Ortho* Team		CRISIS / RAPID RESPONSE		1200	0	1200	
new team		PRIMARY CARE		1200	0	1200	
NightHawk		CASE MANAGEMENT		1200	0	1200	
NightHawk1		CASE MANAGEMENT		1200	0	1200	
NightHawk10		PRIMARY CARE	Homeless	1200	0	1200	
NightHawk11		PRIMARY CARE	Homeless	1200	0	1200	
NightHawk12		PRIMARY CARE	Homeless	1200	0	1200	
NightHawk13		PRIMARY CARE	Homeless	1200	0	1200	
NightHawk14		PRIMARY CARE	Homeless	1200	0	1200	
NightHawk2		CASE MANAGEMENT		1200	0	1200	
NightHawk3		CASE MANAGEMENT		1200	0	1200	
NightHawk4		CASE MANAGEMENT		1200	0	1200	
NightHawk5		CASE MANAGEMENT		1200	0	1200	

Cancel

Alpha8		Active	Open	PRIMARY CARE
Alpha9		Active	Open	PRIMARY CARE
AlphaYa		Active	Open	PRIMARY CARE

This screen provides only one way to search – by Name in the top text box – and all search results are immediately shown in the same dialog box. The user can choose to Cancel, which hides this box and returns the user to the previous screen they were working on, shown masked below the popup.

The embedded table of results has the common set of features described in an earlier section – sorting, paging and filtering.

8.4.3. “Alerts” functionality

When logging in to PCMMR, the user may have one or more Alerts which need addressing. Alerts are simple messages that can either be strictly informational, or require some additional action from the user. The user will see the message “You have one or more active alerts. Please [click here](#) to view them” in a red box next to the PCMMR welcome text. When addressing an informational alert, the user can simply dismiss it in a similar fashion to deleting an email. When addressing an actionable alert, a new red alert box appears across the top of all pages in the user’s current web session signifying they are currently

working an alert, and until they mark the alert as complete, the alert message will continue to be displayed:

CISS Portal Desktop - Mozilla Firefox

File Edit View History Bookmarks Tools Help

CISS Portal Desktop

AFHIL CISS Local PCMM HP Sites VA Sites EEM Connor's HP Room WebLogic 10.3.2 docs

PCMM Patient-Centered Management Module

Station: CHEYENNE VAMC (#442) User Name: [redacted] Log Out

Parent: VISN 19

This is a TEST environment. Do NOT use or enter real Patient Data.

CISS Home OHRS PCMM

Home Patient Management Team Management Staff Reports Alerts Help Reference

PCMM Home Position List

You are currently working on fulfilling alert: One or more positions were created/updated by applying a model team template [Save for Later] [Mark as Complete]

Team Position List

Team Profile:

Name: Nighthawk7 Care Type: PRIMARY CARE Focus: HIV Clinic Status: Active Assignments: Open

Create a Team Position Reconcile with Models

Showing 1 to 2 of 2 entries

Team Role	Description	Staff Role	Staff Name	Preceptor	Preceptee	Status	Expected FTEE	Actual FTEE	Actions
NURSE CARE MANAGER			Not Assigned			Active	1.00		
PSYCHOLOGIST	PCMM Smoke Test		Agent, The			Active	1.00	1.00	

Filters: [dropdown]

First Previous 1 Next Last

Display 25 records

View the Model linked to the Team

If you are experiencing problems/issues with the PCMM software application portlets, please contact the VA Service Desk (VASD) at 1-888-596-4357 to open up a Remedy Ticket. Software issues will not be monitored via the Contact Us link.
If you have a question about the Patient-Centered Management Module (PCMM) Program, please Contact Us.

VA HEALTH CARE Defining EXCELLENCE in the 21st Century

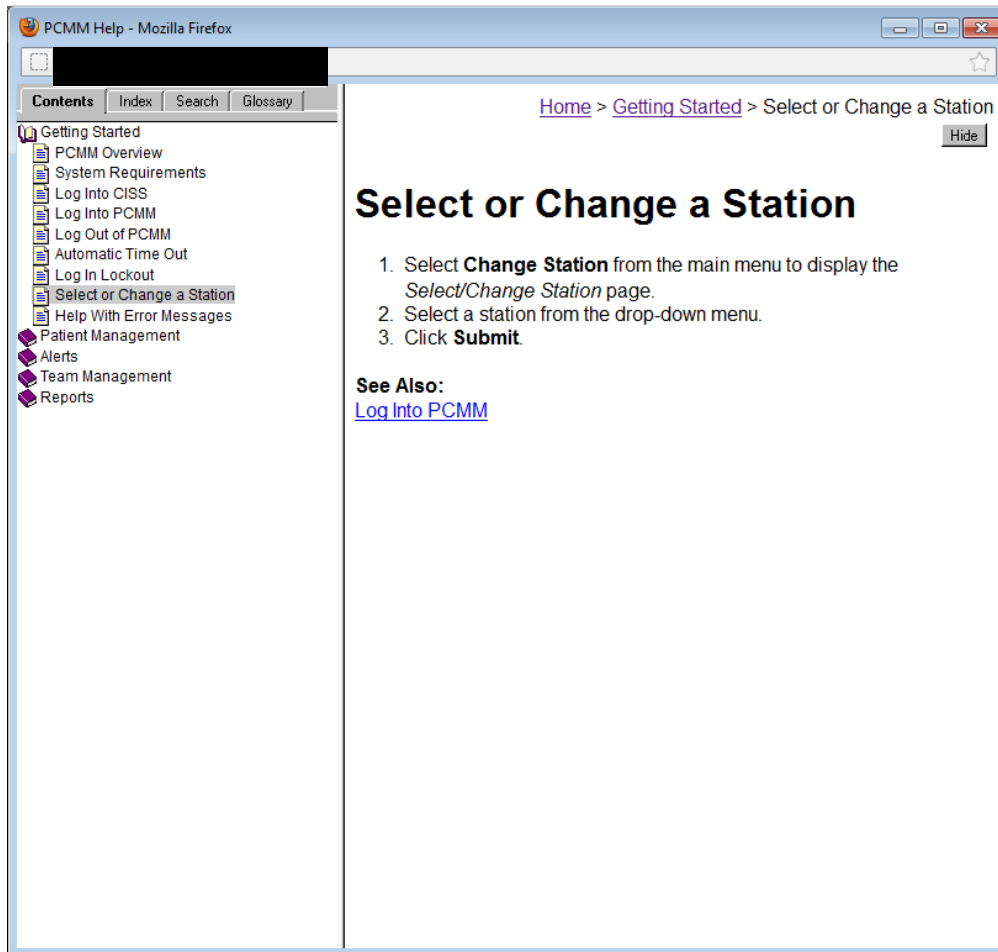
All dates & times are in Mountain Standard Time.
CISS Version: 1.4.3.28.DevX | PCMM Version: 1.0_1248

javascript:updateComments(71)

As shown, the user has two options: “Save for Later”, which removes the red header bar but keeps the alert in their work queue, or “Mark as Complete” meaning the user finished working on the alert and wants to log it as complete.

8.4.4. “Context-Sensitive Help” functionality

PCMMR contains a complete set of context-sensitive help in the standard format displayed below:



This help is accessible on all pages by clicking the blue question mark in the upper left corner, and in most cases, the help will open directly to instructions that pertain to the function the user is performing. In the event that no context-sensitive help exists for the user’s current function, the user will simply be shown the main help table of contents.

This help window opens as a separate window on top of the main PCMMR browser application window, such that the user’s session is not disrupted.

9. System Integrity Controls

All database changes (insertions, deletions and updates) are audited in dedicated history tables, one per main table. The history table naming convention is to append the main table name with an “_H”. These tables contain the user making the change, the date/time the change occurred, the type of change (Insertion, Deletion, Update) and the complete row data before the change occurred. Thus, the complete history of any item in the PCMMR application can be retraced by inspecting the history table entries in the order that they were created.

Additionally, Create/Read/Update/Delete (CRUD) permissions are explicitly defined for all objects throughout the PCMMR application, and assigned to each role in the CISS security level. The current CRUD spreadsheet is in the following document:



PCMM CRUD Matrix

These security restrictions serve to selectively disable or show/hide functions in PCMMR, such as the ability to create new teams, the ability to run various reports, etc. The implementation of these permissions uses the standard Spring security features, and can be activated at both the JSP level:

```
<security:authorize access="hasRole('View Report List')">
    <a href="...">View Report List</a>
</security:authorize>
```

and also the service method level:

```
@PreAuthorize("hasRole('" + CISSPermissionType.UPDATE_TEAM + "')")
public Team updateTeam(Team team, long institutionId) {
    ...
}
```

Access to the database is restricted to a set of users and authentication is performed by SQL Server when logging in (either via the SQL Server Administration tool or over TCP/IP using “SQL Server Authentication” method). Some users who have read-only privileges to the database, such as for reporting, do not have permissions to modify the table structures or data.

10. Appendix A

10.1. Requirements Traceability Matrix



PCMM Requirements
Traceability Matrix

10.2. Packaging and Installation

Per customer requirements, the software will be packaged as a compressed and encrypted ZIP file for delivery. Installation of the software into a production environment involves deploying the application .ear files for both the attended server and unattended servers, the online help and the VistaLink shared libraries and console for connectivity to the various Vista systems.

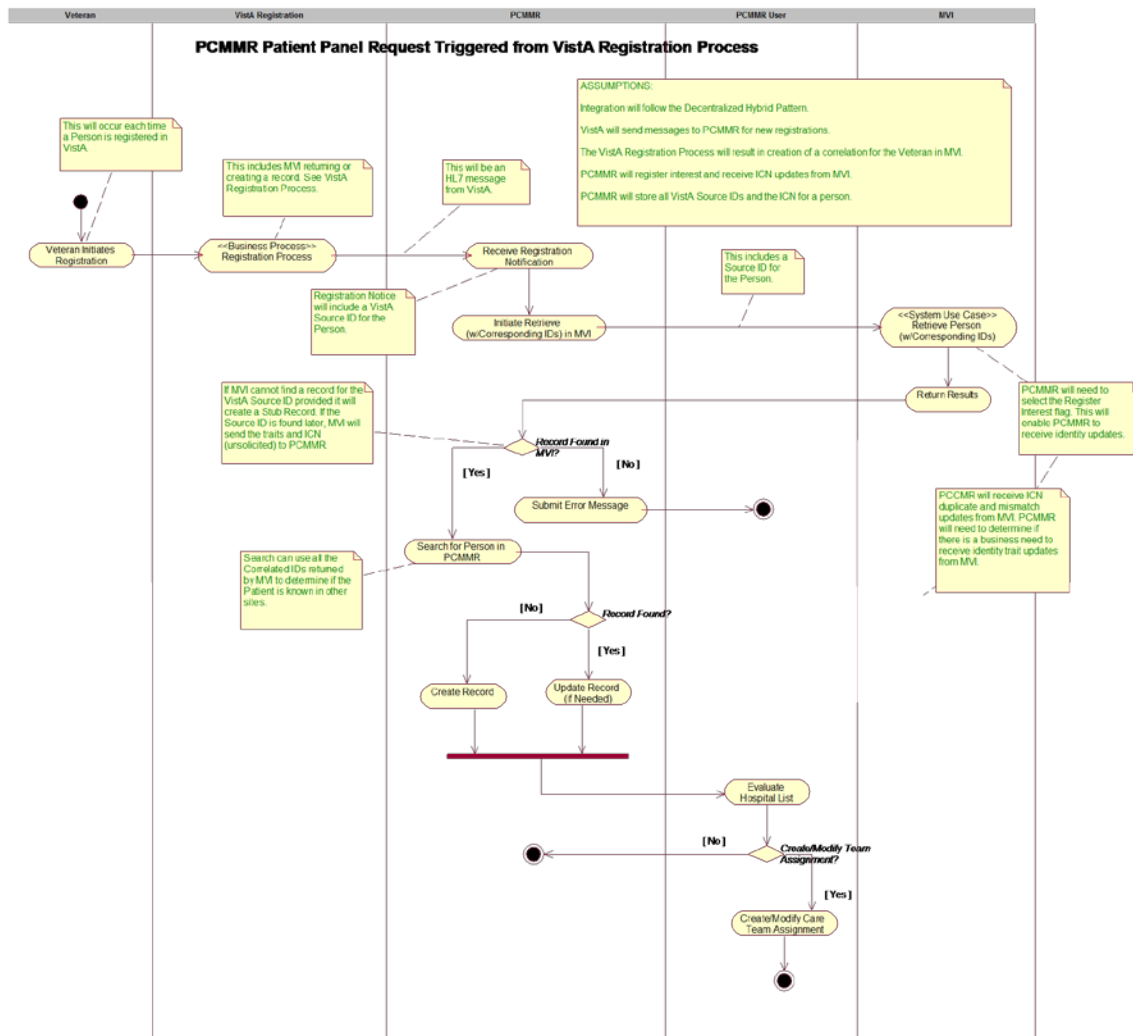
10.3. MVI use case “Manage a Patient Panel Request” requirements

The following requirements break down the logic followed in this use-case:

1. If a VistA Source ID is not available in the initial patient panel request, PCMMR shall use the patient name and Social Security to search VistA for a Source ID.
2. PCMMR shall use a VistA Source ID to send MVI a Retrieve Person with Corresponding IDs request for each initial Patient Panel Request.
3. PCMMR shall pass the “Register Interest Flag” in the Retrieve Person with Corresponding IDs request. This will enable PCMMR to receive identity updates from MVI. PCMMR will receive ICN duplicate and mismatch updates from MVI.

Note: PCMMR will need to determine if there is a business need to receive identity updates from MVI.

- The MVI system will return results to the PCMMR system.
 - If a record is found using the MVI’s Retrieve Person with Corresponding IDs service, it is returned to PCMMR with the active ICN and current Primary View traits for the person queried.
 - If a record is not found, the MVI submits an error message to the PCMMR user and the process ends.
4. If a record is found in MVI, PCMMR shall initiate a Search for Person in the PCMMR system. The Search can use the Source IDs returned by MVI to determine if the Patient is known in other sites.
 5. If a record is found in PCMMR, the PCMMR system shall update the patient record, if needed, with the information provided by MVI.
 6. If a record is not found in PCMMR, PCMMR shall create a record that will include the VistA Source ID and the ICN.
 7. The PCMMR user shall evaluate the patient record and associated list of treating facilities. The PCMMR user will Create or Modify a Team Assignment as needed.



10.4.Design Metrics

The only relevant metrics to take into consideration are those defined in the Requirements Elaboration Document (RED). These metrics describe overall flow in the application and do not necessary specify a detailed screen-by-screen or component-by-component design. Required Technical Documents

The following documents must be submitted for review to support proper approval:

- Product Architecture Document;
- Disaster Recovery Plan;
- Interface Data Mapping
- Security Assurance Strategy

For additional information regarding how to obtain proper approval for this project, refer to the following documents:

- [IT Infrastructure Standards](#)

- [*Systems Engineering and Design Review \(SEDR\) process*](#)
- [*Enterprise Architecture Web page*](#)
- [*One-VA TRM*](#)

Attachment A - Approval Signatures

This section is used to document the approval of the System Design Document (SDD) during the Formal Review. The review should be ideally conducted face to face where signatures can be obtained 'live' during the review however the following forms of approval are acceptable:

1. Digital signatures tied cryptographically to the signer (instructions for digital signature can be found in the Digital Signature Guide in ProPath [REDACTED] [process/Library/digital signature guide.doc](#))
2. Physical signatures obtained face to face or via fax
3. /es/ in the signature block provided that a separate digitally signed e-mail indicating the signer's approval is provided and kept with the document

The Chair of the governing Integrated Project Team (IPT), Business Sponsor, Information Technology (IT) Program Manager, and the Project Manager are required to sign. Please annotate signature blocks accordingly.

