**Computerized Patient Record System**

Enhancments Phase II

System Design Document



**December, 2018**

Version 1.1

Department of Veterans Affairs

Revision History

| Date | Version | Description | Author |
| --- | --- | --- | --- |
| 12/12/18 | 1.2 | Initial addition of NSR 20141108 | Chris Bell |
| 11/30/2018 | 1.1 | Additional updates for NSR 20120407 | Jamie Crumley |
| 11/09/2018 | 1.0 | Initial Draft  NSR 20120407 | Jamie Crumley |

Place latest revisions at top of table.

The Revision History pertains only to changes in the content of the document or any updates made after distribution. It does not apply to the formatting of the template.

Remove blank rows.

Artifact Rationale

The System Design Document (SDD) is a dual-use document that provides the conceptual design as well as the as-built design. This document will be updated as the product is built, to reflect the as-built product.

When to Complete Each Section of the SDD

| Section | Completed On or Before PMAS Phase | Rationale |
| --- | --- | --- |
| 1 – Introduction | MS 0 Review; updated thereafter | Conceptual design should inform evaluation of investments |
| 2 - Background | MS 0 Review; updated thereafter | Conceptual design should inform evaluation of investments |
| 3 – Conceptual Design | MS 0 Review; updated thereafter | Conceptual design should inform evaluation of investments |
| 4 – System Architecture | MS 0 Review; updated thereafter | Conceptual design should inform evaluation of investments |
| 5 – Data Design | MS 1 Review; updated thereafter | Design details should be elaborated upon during PMAS Planning phase and prior to development |
| 6 – Detailed Design | MS 1 Review; updated thereafter | Design details should be elaborated upon during PMAS Planning phase and prior to development |
| 7 – External System Interface Design | MS 1 Review; updated thereafter | Design details should be elaborated upon during PMAS Planning phase and prior to development |
| 8 – Human Machine Interfaces | MS 1 Review; updated thereafter | Design details should be elaborated upon during PMAS Planning phase and prior to development |
| Attachments | MS 1 Review; updated thereafter | Design details should be elaborated upon during PMAS Planning phase and prior to development |

A product’s system design should be defined conceptually prior to the allocation of personnel and resources that occur at project initiation. This gives the enterprise an opportunity to evaluate IT investments before project teams are stood up and funding is allocated. Sections 1- 4 which discuss the high level design should be completed prior to MS 0. All sections should be completed and updated before MS 1. Projects will need to address all SDD approval constraints prior to the MS 2 review. In addition, the SDD should reflect the as-built product going into the MS 2 review.

Instructions

The System Design Document (SDD) is a general purpose document that is use to specify the design of new systems including Custom Development, Software as a Service, Hosted Services; modifications or updates to an existing system. Therefore, all sections may not apply to system being designed. Any sections not applicable to the current effort must be marked N/A along with rational for why the section is not relevant.

This template contains a style named Instructional Text. Text using this style is only to provide guidance in completing the document – the final document should not contain Instructional Text. Text in paragraphs added after Instructional Text is automatically set to the appropriate body text style. For best results and to maintain formatting consistency:

* Use the provided paragraph styles
* Delete all Instructional Text before finalizing the document, including these instructions.

The following project types are required to complete this artifact. Exceptions are outlined where needed throughout the document.

| Activity | New Capability (1) | Feature Enhancement (2) |
| --- | --- | --- |
| **Field Deployment (A)** | Yes | Yes |
| **Cloud/Web Deployment (B)** | Yes | Yes |
| **Mobile Application (C)** | Yes | Yes |

Table of Contents

[1. Introduction 1](#_Toc531377044)

[1.1. Scope 1](#_Toc531377045)

[1.2. User Profiles 1](#_Toc531377046)

[2. Background 1](#_Toc531377047)

[2.1. Overview of the System 1](#_Toc531377048)

[2.2. Overview of the Business Process 1](#_Toc531377049)

[2.3. Overview of the Significant Requirements 1](#_Toc531377050)

[3. Conceptual Design 2](#_Toc531377051)

[3.1. Conceptual Application Design 2](#_Toc531377052)

[3.1.1. Application Context 2](#_Toc531377053)

[3.1.2. High-Level Application Design 5](#_Toc531377054)

[3.1.3. Application Locations 8](#_Toc531377055)

[3.2. Conceptual Data Design 9](#_Toc531377056)

[3.2.1. Project Conceptual Data Model 9](#_Toc531377057)

[3.2.2. Database Information 9](#_Toc531377058)

[3.2.3. User Interface Data Mapping 9](#_Toc531377059)

[3.2.3.1. Application Screen Interface 10](#_Toc531377060)

[3.2.3.1.1. <Insert name of screen> 10](#_Toc531377061)

[3.2.3.2. Application Report Interface 10](#_Toc531377062)

[3.2.3.2.1. <Insert name of report> 10](#_Toc531377063)

[3.2.3.3. Unmapped Data Element 11](#_Toc531377064)

[3.3. Conceptual Infrastructure Design 11](#_Toc531377065)

[3.3.1. System Criticality and High Availability 11](#_Toc531377066)

[3.3.2. Special Technology 12](#_Toc531377067)

[3.3.3. Technology Locations 12](#_Toc531377068)

[3.3.4. Conceptual Infrastructure Diagram 13](#_Toc531377069)

[3.3.4.1. Location of Environments and External Interfaces 13](#_Toc531377070)

[3.3.4.2. Conceptual Production String Diagram 15](#_Toc531377071)

[4. System Architecture 15](#_Toc531377072)

[4.1. Hardware Architecture 15](#_Toc531377073)

[4.2. Software Architecture 15](#_Toc531377074)

[4.3. Network Architecture 16](#_Toc531377075)

[4.4. Service Oriented Architecture / ESS 16](#_Toc531377076)

[4.5. Enterprise Architecture 17](#_Toc531377077)

[5. Data Design 17](#_Toc531377078)

[5.1. DBMS Files 17](#_Toc531377079)

[5.2. Non-DBMS Files 17](#_Toc531377080)

[5.3. Data View 18](#_Toc531377081)

[6. Detailed Design 18](#_Toc531377082)

[6.1. Hardware Detailed Design 18](#_Toc531377083)

[6.2. Software Detailed Design 19](#_Toc531377084)

[6.2.1. Conceptual Design 19](#_Toc531377085)

[6.2.1.1. Product Perspective 19](#_Toc531377086)

[6.2.1.1.1. User Interfaces 19](#_Toc531377087)

[6.2.1.1.2. Hardware Interfaces 19](#_Toc531377088)

[6.2.1.1.3. Software Interfaces 19](#_Toc531377089)

[6.2.1.1.4. Communications Interfaces 20](#_Toc531377090)

[6.2.1.1.5. Memory Constraints 20](#_Toc531377091)

[6.2.1.1.6. Special Operations 20](#_Toc531377092)

[6.2.1.2. Product Features 20](#_Toc531377093)

[6.2.1.3. User Characteristics 20](#_Toc531377094)

[6.2.1.4. Dependencies and Constraints 20](#_Toc531377095)

[6.2.2. Specific Requirements 21](#_Toc531377096)

[6.2.2.1. Database Repository 21](#_Toc531377097)

[6.2.2.2. System Features 22](#_Toc531377098)

[6.2.2.3. Detailed Design by NSR 22](#_Toc531377099)

[6.2.2.3.1. NSR 20120407: Two New Health Summary Objects 22](#_Toc531377100)

[6.2.2.4. Design Element Tables 39](#_Toc531377101)

[6.2.2.4.1. Routines (Entry Points) 40](#_Toc531377102)

[6.2.2.4.2. Templates 41](#_Toc531377103)

[6.2.2.4.3. Bulletins 42](#_Toc531377104)

[6.2.2.4.4. Data Entries Affected by the Design 43](#_Toc531377105)

[6.2.2.4.5. Unique Record(s) 44](#_Toc531377106)

[6.2.2.4.6. File or Global Size Changes 44](#_Toc531377107)

[6.2.2.4.7. Mail Groups 44](#_Toc531377108)

[6.2.2.4.8. Security Keys 45](#_Toc531377109)

[6.2.2.4.9. Options 47](#_Toc531377110)

[6.2.2.4.10. Protocols 49](#_Toc531377111)

[6.2.2.4.11. Remote Procedure Call (RPC) 50](#_Toc531377112)

[6.2.2.4.12. Constants Defined in Interface 51](#_Toc531377113)

[6.2.2.4.13. Variables Defined in Interface 51](#_Toc531377114)

[6.2.2.4.14. Types Defined in Interface 51](#_Toc531377115)

[6.2.2.4.15. GUI 52](#_Toc531377116)

[6.2.2.4.16. GUI Classes 52](#_Toc531377117)

[6.2.2.4.17. Current Form 52](#_Toc531377118)

[6.2.2.4.18. Modified Form 52](#_Toc531377119)

[6.2.2.4.19. Components on Form 52](#_Toc531377120)

[6.2.2.4.20. Events 53](#_Toc531377121)

[6.2.2.4.21. Methods 53](#_Toc531377122)

[6.2.2.4.22. Special References 53](#_Toc531377123)

[6.2.2.4.23. Class Events 53](#_Toc531377124)

[6.2.2.4.24. Class Methods 53](#_Toc531377125)

[6.2.2.4.25. Class Properties 53](#_Toc531377126)

[6.2.2.4.26. Uses Clause 53](#_Toc531377127)

[6.2.2.4.27. Forms 53](#_Toc531377128)

[6.2.2.4.28. Functions 54](#_Toc531377129)

[6.2.2.4.29. Dialog 56](#_Toc531377130)

[6.2.2.4.30. Help Frame 57](#_Toc531377131)

[6.2.2.4.31. HL7 Application Parameter 58](#_Toc531377132)

[6.2.2.4.32. HL7 Logical Link 59](#_Toc531377133)

[6.2.2.4.33. COTS Interface 59](#_Toc531377134)

[6.3. Network Detailed Design 60](#_Toc531377135)

[6.4. Security and Privacy 60](#_Toc531377136)

[6.4.1. Security 60](#_Toc531377137)

[6.4.2. Privacy 60](#_Toc531377138)

[6.5. Service Oriented Architecture / ESS Detailed Design 61](#_Toc531377139)

[6.5.1. Service Description for <Consumed Service Name> 61](#_Toc531377140)

[6.5.2. Service Design for <Provided Service Name> 61](#_Toc531377141)

[6.5.2.1. Introduction 61](#_Toc531377142)

[6.5.2.1.1. Purpose and Scope of Service 61](#_Toc531377143)

[6.5.2.1.2. Links to Other Documents 61](#_Toc531377144)

[6.5.2.2. Service Details 62](#_Toc531377145)

[6.5.2.2.1. Service Identification 62](#_Toc531377146)

[6.5.2.2.2. Service Versions 63](#_Toc531377147)

[6.5.2.2.3. Summary of Design and Platform Details 63](#_Toc531377148)

[6.5.2.2.3.1. SOA Pattern(s) Implemented 63](#_Toc531377149)

[6.5.2.2.3.2. COTS Platform vendor names and versions for hosting platform 63](#_Toc531377150)

[6.5.2.3. Dependencies 63](#_Toc531377151)

[6.5.2.4. Service Design Details 63](#_Toc531377152)

[6.5.2.4.1. Interface Technical Specs 63](#_Toc531377153)

[6.5.2.4.1.1. Service Invocation Type 64](#_Toc531377154)

[6.5.2.4.1.2. Service Interface Type 64](#_Toc531377155)

[6.5.2.4.1.3. Service Name 64](#_Toc531377156)

[6.5.2.4.1.4. Interface 64](#_Toc531377157)

[6.5.2.4.1.5. End Points 64](#_Toc531377158)

[6.5.2.4.1.6. Operations or Methods 64](#_Toc531377159)

[6.5.2.4.1.7. Message Schemas 64](#_Toc531377160)

[6.5.2.4.2. Information Model 65](#_Toc531377161)

[6.5.2.4.2.1. Class Diagram and Description of Entities Involved 65](#_Toc531377162)

[6.5.2.4.2.2. Mappings from ELDM to Standards Based Schemas 65](#_Toc531377163)

[6.5.2.4.3. Behavior Model (AKA Use Case Realization) 65](#_Toc531377164)

[6.5.2.4.3.1. Use Cases (Use Case Model) 65](#_Toc531377165)

[6.5.2.4.3.2. Interaction Diagrams 65](#_Toc531377166)

[6.5.2.5. Gap Analysis 65](#_Toc531377167)

[6.5.2.5.1. Variances from Enterprise Target Architecture 66](#_Toc531377168)

[6.5.2.5.2. Variances from SLDs 66](#_Toc531377169)

[6.5.2.5.3. Variances from Standards and Policies 66](#_Toc531377170)

[6.5.2.5.4. Justification for Exceptions and Mitigation 66](#_Toc531377171)

[7. External System Interface Design 66](#_Toc531377172)

[7.1. Interface Architecture 66](#_Toc531377173)

[7.2. Interface Detailed Design 67](#_Toc531377174)

[8. Human-Machine Interface 68](#_Toc531377175)

[8.1. Interface Design Rules 68](#_Toc531377176)

[8.2. Inputs 68](#_Toc531377177)

[8.3. Outputs 68](#_Toc531377178)

[8.4. Navigation Hierarchy 69](#_Toc531377179)

[8.4.1. Screen [x.1] 69](#_Toc531377180)

[8.4.2. Screen [x.2] 69](#_Toc531377181)

[8.4.3. Screen [x.3] 69](#_Toc531377182)

[9. Attachment A – Approval Signatures 69](#_Toc531377183)

[A. Additional Information 70](#_Toc531377184)

[A.1. Identification of Technology and Standards 70](#_Toc531377185)

[A.2. Constraining Policies, Directives and Procedures 70](#_Toc531377186)

[A.3. Requirements Traceability Matrix 70](#_Toc531377187)

[A.4. Packaging and Installation 70](#_Toc531377188)

[A.5. Design Metrics 70](#_Toc531377189)

# Introduction

The purpose of this System Design Document (SDD) is to cover the Computerized Patient Record System (CPRS) Enhancements Phase II (EP2) project. The CPRS EP2 project includes multiple independent releases, as well as a release for the CPRS Graphical User Interface (GUI) application and associated VistA patches.

## Scope

Requirements associated with CPRS EP2 are stored in the Rational © tool. The first scheduled release will be for NSR 20120407 and requirements are here: [NSR 20120407 Release Collection](https://clm.rational.oit.va.gov/rm/web#action=com.ibm.rdm.web.pages.showArtifact&artifactURI=https%3A%2F%2Fclm.rational.oit.va.gov%2Frm%2Fresources%2F_OTAUsdOvEeisc-MNbXOZPw&vvc.configuration=https%3A%2F%2Fclm.rational.oit.va.gov%2Frm%2Fcm%2Fstream%2F_PWydlh6uEeafn8RFT4tzpQ&componentURI=https%)

## User Profiles

There are several types of users who utilize CPRS. Primarily, the intended users are providers such as physicians, nurses, repiratory therapists, pharmacists, and clergy who are dealing with patient care. In addition, there are laboratorians, financial staff, and others who use CPRS to find patient-related information or enter progress notes related to patients.

Describe the intended user base of the proposed system.

Describe the attributes of the user community (and their proficiency with software systems) and the technical community (and their familiarity with support and maintenance).

# Background

## Overview of the System

Provide a brief overview of the system in business terms, i.e.

* Explain what the system does?
* The benefits of developing the system.

This section should identify:

* The participants in its operation, and
* The functions that they perform.
* And explain the role of parties external to the Office of Information and Technology (OIT).

## Overview of the Business Process

Provide an overview of the business processes that this application will support. Provide a link to the RSD.

## Overview of the Significant Requirements

Provide a link to the RSD and the BRD. Provide a link to a Rational report which can provide requirements for the following:

* Overview of significant functional requirements
* Overview of the functional workload/performance requirements
* Overview of operational requirements
* Overview of pivotal technical requirements
* Overview of the security and privacy requirements
* Overview of system criticality and high availability requirements
* Overview of single sign on requirements
* Overview of use of enterprise portals
* Overview of special device requirements

# Conceptual Design

This section of the SDD provides details about the following topics:

* Conceptual Application Design
* Conceptual Data Design
* Conceptual Infrastructure Design

## Conceptual Application Design

Provide details of the ‘As-is” view of the existing system along with the design of “the current increment” and the “To-be.”. The “To-Be” view should include the future application context, and application high level design. The “current increment” view should have application context and high level design of this specific increment that this SDD addresses.

### Application Context

.

Provide a diagram showing the context within which the application exists. The diagram should include:

* One object for the system that is the subject of this design,
* One object for each system or external service with which this system interfaces,
* One object for each Program Office system or subsystem with which this system interacts, and
* One for each data store that this system shares with other systems.

Sample Application Context Diagram

Sample Application Context Diagram

Sample Application Context Diagram

Figure 1: Sample Application Context Diagram

Table 5 describes the information in the Application Context Diagram in four sections. Note that the system for which this design applies is represented by a single object (typically in the center of the diagram). Therefore, it is not referred to in Table 5 below.

Table 5 (Grouping): Application Context Description

Object

| ID | Name | Description | Interface Name | Interface System |
| --- | --- | --- | --- | --- |
| < ID from diagram> | <Enter name of external system, organization, or agency> | <High level discussion of the purpose of the information interchange> | <Name of each of the Interfaces to this object> | <Systems with which this system interfaces> |

Interfaces External to OI&T

| ID | Name | Related Object | Input Messages | Output Messages | External Party |
| --- | --- | --- | --- | --- | --- |
| < ID from diagram> | <Interface name from the object rows above> | <Object from the list above that is the source of this interface> | <For each input message, enter a business description of the data being input> | <For each output message, enter a business description of the data being output> | <Name of external party> |

Interfaces Internal to OI&T

| ID | Name | Related Object | Input Messages | Output Messages | External Party |
| --- | --- | --- | --- | --- | --- |
| < ID from diagram> | <Interface name from the object rows above> | <Object from the list above that is the source of this interface> | <For each input message, enter a business description of the data being input> | <For each output message, enter a business description of the data being output> | <Name of external party> |

Externally Shared Data Stores

| ID | Name | Data Stored | Owner | Access |
| --- | --- | --- | --- | --- |
| < ID from diagram> | <Name of the data store> | <Description of the data being stored> | <This System / Name of OIT or external organization> | <Enter the Create, Read, Update, or Delete (CRUD) operations that this system does on this data store> |

### High-Level Application Design

The High-Level Application Design identifies the major components of the application and the relationships of the major application components to each other and to the surrounding applications. The major components of the application are at the subsystem or top-level service area. Many different graphical formats are acceptable for the High-Level Application Design Diagram. Lower-level services will be defined and documented in the Logical Application Design section.

**Error! Reference source not found.** illustrates a High-Level Application Design in the form of a dataflow diagram. This diagram differs from the diagram in Figure 2 in that the single object representing this system in Figure 2 is decomposed into its major components. Use

Table 6 to describe the objects in **Error! Reference source not found.**.

Note: If an extension to a legacy system is being developed without use of services, all references to “Service” should be changed to “Subsystem.”

A Collaboration Diagram, or in the case of Services, a Service Capability Diagram may be included instead or an Application Diagram if it illustrates the subject better.

Sample High-Level Application Design

Sample High-Level Application Design

Sample High-Level Application Design

Figure 2: Sample High-Level Application Design

Table 6: Objects in the High Level Application Design

Objects / Components to be Built or Modified

| ID | Name | Description | Service or Legacy Code | External Interface Name | External Interface ID | Internal Interface Name | Internal Interface ID | SDP Sections 1&2 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| < ID from diagram> | <Name of high level service or internal subsystems> | <Business level discussion of the function or role of the service or subsystem> | <Service / modification to legacy system> | <Name of each of the external interfaces to this object> | <ID of each of the external interfaces to this object> | <Name of each of the internal interfaces to this object> | <ID of each of the internal interfaces to this object> | [Approved / Submitted / Being Developed] |

Internal Data Stores

| ID | Name | Data Stored | Steward | Access |
| --- | --- | --- | --- | --- |
| < ID from diagram> | <Name of the data store> | <Description of the data being stored> | <Name of the system/subsystem /service that is the steward for the data> | <Which CRUD operations does this system do on this data store> |

### Application Locations

Use Table 7 to specify the locations at which the application components will be hosted.

Consideration should be given to adopt cloud technologies as potential solutions. Leveraging cloud technologies is part of a larger effort by the Office of Management and Budget (OMB) to reform Federal IT Management. Considerations such as regional deployments etc. should be documented in this section.

Table 7: Application Locations

| Application Component | Description | Location at Which Component is Run | Type |
| --- | --- | --- | --- |
| <Component name> | <Description> | <Facility name> | <Presentation Logic/Business Logic/Data Logic/Interface Code> |

Table 8: Application Users

| Application Component | Location | User |
| --- | --- | --- |
| <Component name> | <Facility name> | <Role> |

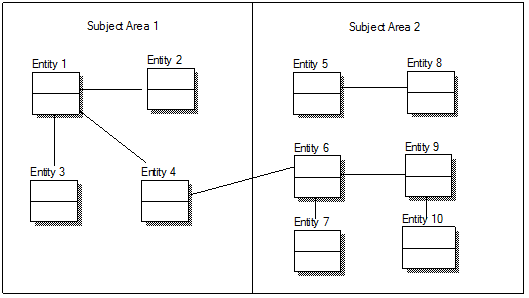
## Conceptual Data Design

### Project Conceptual Data Model

A project conceptual data model (CDM) is a high-level representation of the data entities and their relationships. It does not normally include the data elements that comprise each entity. It is a first step toward developing the more detailed logical data model (LDM) that will be provided during the Logical Data Design.

Figure 3 illustrates a sample of a project CDM.

Figure 3: Sample Project Conceptual Data Mode

****

### Database Information

Use Table 9 to identify all the databases that will be created, replaced, interfaced with, or whose structure will be modified (i.e., add or delete tables or add or delete columns to a table) as part of this effort.

Table 9: Database Inventory

| Database Name | Description | Type | Steward |
| --- | --- | --- | --- |
| <Name> | <Description> | <Create/Replace/Interface /Modify> | <Application/Organization that is the steward> |

### User Interface Data Mapping

This section describes and defines the format and information that will be available for users of the product to be able to enter data into the database or to retrieve information from the database, if applicable.

#### Application Screen Interface

Create a new subsection for each screen of the Graphical User Interface (GUI) that users will have access to, in order to enter or update information in the database.)

##### *<Insert name of screen>*

Figure 4: <screen name> Screen represents the screen that <describes what the screen accomplishes>; Table 10 describes it. Paste a screenshot below and complete the table to describe the screen.

Figure 4: *<screen name>* Screen

Table 10: *<screen name>* Screen Description

| Graphical User Interface (GUI) Field | Table (Database Table that field connects to) | Field (Field in Table that the GUI field connects to) | Comments |
| --- | --- | --- | --- |
| <Name> | <xxx> | <PATIENT\_ NAME> | <Add any comments or descriptive information that would be relevant to the tester> |
| <SSN> | <xxx> | <SSN> |  |
| Date of Birth (Age) | <yyyy> | DATE\_OF\_BIRTH DATE\_OF\_DEATH (if deceased) |  |

#### Application Report Interface

This section describes and defines the reports that will be available in the user interface, if applicable.

##### *<Insert name of report>*

<Create a new subsection for each report> Figure 6 represent <name> screen and Table 16 describes it…

Figure 5 represents the <report name>; Table 11 describes it. Paste a screenshot of the report below and complete the table to describe the report.

Figure 5: *< Report name>* Report

Table 11: *<Report name>* Description

| Report Column | Data Source *<Table Name. Fieldname>* |
| --- | --- |
| Patient | <xxx.PATIENT\_NAME> |
| SSN | <xxx.SSN> |
| DoB | <yyyy.DATE\_OF\_BIRTH> |

#### Unmapped Data Element

In this section describe any database element that was not mapped to a screen and the reason the data element(s) was not mapped. This section may be skipped if there is no User Interface involved in the project, such a building a service offering etc.

## Conceptual Infrastructure Design

The Conceptual Infrastructure Design should describe any unique technology that will be used, which are either part of this system, or will attach to this system.

. Because the system is at a preliminary design stage, it is expected that the information provided may need to be changed during later design stages or increments.

The Conceptual Infrastructure Design is a high-level overview of the infrastructure that will be used to support the application. Primary emphasis is on the environments that will be required and the locations at which they will be installed. The Conceptual Infrastructure Design becomes more detailed at later stages as more information is collected regarding the system, and the infrastructure requirements (i.e., capacity requirements) are better known.

### System Criticality and High Availability

Describe the approach that will be taken to meeting the system criticality and high availability requirements identified in Section 2.5.6, including the extent to which geographically distributed, high availability designs are planned. Describe the approach that is taken towards high availability as well as any workload distribution scheme that is planned to support the high availability implementation (e.g., restricting updates to a single node).

If the system is not mission critical and high availability is not required, then describe the approach that will be taken to provide the requisite level of availability and disaster recovery.

### Special Technology

If any special technology was identified in Section 2.5.9 as part of this system, describe the device and the type of location at which it will be installed. This information may be provided using Table 12.

Table 12: Special Technology Requirements

| Special Technology | Description | Notional Location | TRM Status |
| --- | --- | --- | --- |
| <Name> | <Business language description> | <At what type of location will this technology be deployed?> | <Is this technology in the TRM?  (Yes / No)> |

### Technology Locations

This section describes the various technology components that will be used. If known, provide the name of the datacenter at which the technology will be installed. If not, specify as Site A, Site B etc. Provide this information in Table 13.

Table 13: Technology Location Details

| Technology Component  Production 1 | Location | Usage |
| --- | --- | --- |
| Workstations |  |  |
| Special Hardware |  |  |
| Interface Processors |  |  |
| Legacy Mainframe |  |  |
| Legacy Application Server |  |  |
| Legacy Databases |  |  |
| Other |  |  |

| Technology Component  Production 2 | Location | Usage |
| --- | --- | --- |
| <copy from Prod 1 set, or enter new ones as appropriate> |  |  |

| Technology Component  Certification | Location | Usage |
| --- | --- | --- |
|  |  |  |

| Technology Component  Education | Location | Usage |
| --- | --- | --- |
|  |  |  |

| Technology Component  Test | Location | Usage |
| --- | --- | --- |
|  |  |  |

| Technology Component  Development | Location | Usage |
| --- | --- | --- |
|  |  |  |

### Conceptual Infrastructure Diagram

#### Location of Environments and External Interfaces

Create a diagram to show the environments that will be supported. As illustrated in

Figure 6, the diagram should show the following:

* Local networks to which they will be attached (Production, Test, or Development)
* Locations at which they will be installed
* External connections (each external interface should be shown in terms of where it enters the network).

Sample Conceptual Networks and Environments

Sample Conceptual Networks and Environments

Sample Conceptual Networks and Environments

Figure 6: Sample Conceptual Networks and Environments

#### Conceptual Production String Diagram

Create a diagram to show the configuration of a single production string.

Additional components, such as the mainframe, other Web servers, or other major components should be included if they are expected to be required.

Figure 7: Conceptual Production String Diagram

# System Architecture

This section describes the system and/or subsystem(s) architecture for the project. Discuss the general architectural decisions that have been approved. Include diagrams where appropriate.

## Hardware Architecture

Describe the system hardware architecture and indicate whether the processing system is distributed or centralized. List and describe the hardware modules with diagrams showing the connectivity between the modules. If possible, identify the type, number, and location of servers, workstations, processors, backup systems, and output devices. Include information related to the capacity planning of the system.

## Software Architecture

Describe the overall system software and organization. List and describe the software modules (i.e., including functions, subroutines, or classes), programming languages, and development tools.

Describe all software required to support the system, and specify the physical location of all software systems. Identify database platforms, compilers, utilities, operating systems, and communications software.

Provide diagrams that illustrate the segmentation levels down to the lowest level. Include names and reference numbers for all features on the diagrams. Include a narrative that expands on and enhances the understanding of the functional breakdown.

Note: Diagrams should map to the Requirements Specification Document’s data flow diagrams.

## Network Architecture

Describe communications within the system, such as local area networks (LANs) and buses. Include the communications architecture(s) being implemented, such as X.25 and token ring.

Provide a diagram depicting the communications path(s) between the system and subsystem modules.

## Service Oriented Architecture / ESS

This subsection of the SDD should put the product into perspective with other related products. This is achieved in the high level design.

* If the product is independent and totally self-contained, it should be so stated here.
* If the SDD defines a product that is a component of a larger system, as occurs frequently, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software. It is highly recommended that the SDD and other related artifacts of the larger system are included by reference, with links and not duplicate huge chunks of it here, which could potentially get out of sync. Integration projects depend on all parties understanding the same things about their relationships, and such information should be in one document and referenced by link as needed.

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful.

Services Provided: Those shared services that will be provided as part of this application (if the project is a combined solution and service development project). The Data Exchanges should then be included as part of whatever service is providing them. This may also be described as an attribute of the components listed in the high level application design when appropriate.

Service Required/Consumed: This would be the services this solution/application depends on. Again, data exchanges should be included as part of the service descriptions. This should also be adequately described in the conceptual and integration sections as appropriate.

Provide a diagram depicting the Enterprise Shared Services between the system and subsystem modules.

If the system currently being built is in-flight or in-transition, then depict the as-is, interim and target states of the system with diagrams, and identify the Enterprise Shared Services consumed or provided. This will be part of the conceptual solution design.

If the solution proposed is a duplication of an existing service, or a stand-alone silo solution, then appropriate justification needs to be provided.

## Enterprise Architecture

Describe the Enterprise Architecture of the system.

Show adherence to the VA Technical Reference Model (TRM)/ Standards Profile (SP). New system development and selection must adhere to approved standards and rules, unless it proves to be more cost-effective over the life of the application to deviate from the standards. The standards, strategies, and guidelines establish the fundamental technologies enabling the VA to meet many of its business and information system goals. By using these standards, the VA can promote interoperability, portability and adaptability within systems, promote quality assurance, place the VA in a position to utilize current technology, and provide a framework for IT application and infrastructure development. The current TRM/SP is located VA Enterprise Architecture (EA) v2.1 at <http://DNS.URL/ToolRequestPage.aspx?treqid=34239>.

# Data Design

This section outlines the design of the database management system (DBMS) and non-DBMS files associated with the system. For networks, detail the distribution of data and identify any changes to the logical data model that may occur due to software or hardware requirements.

Note: Provide a data dictionary appendix showing data element name, type, length, source, validation rules, maintenance, data stores, outputs, aliases, and description.

## DBMS Files

If a database will be used list and describe the logical requirements that exist for data formats, storage capabilities, data retention, data integrity, etc.

Describe how the database will be designed, including the following information, as appropriate:

* Logical model; provide normalized table layouts, entity relationship diagrams, and other logical design information
* DBMS schemas, subschemas, records, sets, tables, storage page sizes
* Access methods (such as indexed, via set, sequential, random access, sorted pointer array)
* Estimate the database file size or volume of data within the file, data pages, including overhead resulting from access methods and free space
* Definition of the update frequency of the database tables, views, files, areas, records, and sets
* Estimates on the number of transactions that the database may have to process.

## Non-DBMS Files

* Describe all non-DBMS files including narratives on the usage of each file.
* Identify if the file is used for input, output, or both; identify temporary files, which modules read and write the file, and similar.
* Identify record structures, record keys, indices, and reference data elements within the records.
* Define record length and blocking factors.
* Define the file access method such as: index sequential, virtual sequential, random access.
* Estimate the file size or volume of data within the file.
* Define the update frequency of the file if appropriate. Provide the estimated number of transactions per unit time and the statistical mean, mode, and distribution of those transactions.

## Data View

A "Data View" should be included in the Architectural Representation whenever persistent data objects are included in the system (they are typically present in most software systems). The data view describes the logical data model of the system and includes an Entity Relationship Diagram (ERD). For a description of Entity Relationship diagramming please refer to the whitepaper <<http://www-106.ibm.com/developerworks/rational/library/content/03July/2500/2785/2785_uml.pdf>>

# Detailed Design

This section describes the proposed design in detail. Provide the necessary information for the development team to integrate the hardware components and write the software code, so that the hardware and software components will provide a functional product. This is the detailed design, based upon the conceptual design (high level) that was described in the document up to this point.

Note: Every design item should map back to the Requirements Specification Document. These should be captured in the Requirement Traceability Matrix (RTM).

## Hardware Detailed Design

The information requested in this section may be provided by Engineering and/or the Developers. The information provided here is mainly for use by Engineering and Operations.

In this section, provide enough information for the developers to build and/or procure the system’s hardware. The level of detail requested should be treated as a general guideline and can be omitted if it needs to be filled in by Engineering and Operations.

Note: If this section becomes too lengthy, consider incorporating it as an appendix or reference it in a separate document. Add additional diagrams, if necessary, to describe each component and its functions.

Include the following information (as applicable):

* How much compute capacity? (MFLOPS, TPMs etc.)
* System Memory
* Local and Shared storage
* Network requirements (Bandwidth, Latency etc.)
* Public or Private cloud

## Software Detailed Design

This section provides conceptual and final detailed information associated with the design of the software being delivered. This should be an extension of the corresponding section from Section 3.1, but should contain additional detail as the project progresses.

### Conceptual Design

This section introduces the conceptual information that establishes the basis for how the software will be built.

#### Product Perspective

This subsection of the SDD should put the product into perspective with other related products. If the product is independent and completely self-contained, it should be stated here. If the SDD defines a product that is a component of a larger system, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful.

Sections of the Requirements Specification Document (RSD) can be referenced in the subsections, if applicable.

##### User Interfaces

This subsection should specify the logical characteristics of each interface between the software product and its users. This includes those configuration characteristics necessary to accomplish the software requirements (e.g., screens, roll and scroll, GUI interface).

Recommendation: Create a block diagram showing the user interfaces.

##### Hardware Interfaces

This subsection should specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics (for example, hardware platform or mainframe versus personal computer). It also covers matters such as what devices the system will support, how they will be supported, and protocols. Examples include scanners, pen driven devices, and radio frequency devices.

Recommendation: Create a block diagram showing the hardware interfaces.

##### Software Interfaces

This subsection should specify the use of other required software products (e.g., VA Kernel, VA FileMan, Windows NT); and interfaces with other applications or other systems such as commercial off-the-shelf (COTS) or national databases. Specify the application interfaces (e.g., the linkage between an accounts receivable system and a general ledger system and a COTS software package that will be interfaced using an existing interface). This section should provide the following information for each required software product:

* Name
* Version number
* Discussion of the purpose of the interfacing software as related to this software product
* Definition of the interface in terms of message content and format (e.g., Health Level Seven [HL7], electronic data interchange).

##### Communications Interfaces

This subsection should specify the various interfaces to communications such as local network protocols, e-mail, Transmission Control Protocol (TCP), modems.

Recommendation: Create a block diagram showing the communications interfaces.

##### Memory Constraints

This subsection should specify any applicable characteristics and limits on memory or partition size.

##### Special Operations

This subsection should specify the special operations required by the user such as backup, recovery, and archiving operations.

This section should also include any operations for external devices or COTS systems.

#### Product Features

This subsection should provide a summary of the major features of the software.

For example, an SDD for an accounting program might use this section to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those features requires.

Note: For clarity, remember these items when creating this section of the SDD:

* The features should be organized in a way that makes the list of features understandable to the customer or to anyone else reading the document for the first time.
* Textual or graphical methods can be used to show the different features and their relationships.
* Such a diagram is not intended to show a design of a product, but simply shows the logical relationships among variables.

#### User Characteristics

This subsection should describe the general characteristics of the intended users of the product, including experience and technical expertise. It should not be used to state specific requirements but rather should provide the reasons why certain specific requirements are specified in the RSD.

#### Dependencies and Constraints

This subsection should provide a description of any other items that will limit the developer’s options. The following list includes items that limit the developer’s options.

* Regulatory policies
* Hardware limitations (for example, signal timing requirements)
* Interfaces to other applications
* Parallel operation
* Audit functions
* Control functions
* Higher-order language requirements
* Reliability requirements
* Criticality of the application
* Safety and security considerations
* Usability (including 508 compliance)

This section of the SDD should contain all the software design to a level of detail sufficient to enable programmers to develop a system that satisfies the requirements defined in the RSD. It should be detailed so as to make it easy for technical staff to find the methods to complete the designed function.

These requirements should, at minimum, include the following items:

* An indication of the associated requirement(s) in the RSD which is being designed
* A description of the functionality being designed
* The design entities (and their attributes) affected
* The algorithm executed (where appropriate) to implement the functionality.

Because the Dependencies and Constraints section is often the largest and most important part of the SDD, the following principles apply:

* Specific design should be cross-referenced to earlier, related documents (e.g., the RSD).
* All design should be uniquely identifiable.
* Items in this section should be identified from a technical level rather than an end user level. (i.e., an option name should be identified rather than the menu text for that option).

### Specific Requirements

#### Database Repository

The Database Repository section in the RSD can be referenced in this section.

If a logical database design is a part of the system, it should be listed here. Logical database design should specify the logical requirements for any information that is to be placed into a database. This may include:

* Types of information used by various functions
* Frequency of use
* Accessing capabilities
* Data entities and their relationships
* Integrity constraints
* Data retention requirements.

Recommendation: Create a block diagram showing the databases and where the data resides.

#### System Features

Describe the system features, functional requirements, sub-requirements, etc. which can be organized in an outline format that matches the RSD. Specific formatting and organization of the paragraphs (i.e., section numbering) is left to the discretion of the author and is dependent on the level of detail essential to fully describe the design. Some designs may only require two levels; others may require multiple levels. The information necessary to define the items or to specify modifications to the items affected by the functionality being designed should be provided in the appropriate design element tables. Where feasible, instead of duplicating the RSD, it can be referenced via a link, to avoid unnecessary duplication. The key goal is to provide traceability to requirements.

#### Detailed Design by NSR

##### NSR 20120407: Two New Health Summary Objects

The release for this NSR consists of three patches: GMTS\*2.7\*125, PSS\*1\*234, and PSN\*4\*567

Table 15 GMTSADH5

| Routines | Activities | | | |
| --- | --- | --- | --- | --- |
| **Routine Name** | GMTSADH5 | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** | RM Artifacts: 1041110, 1041111, 1041112, 1041113, 1041114, 1041115, 1041116 | | | |
| **Related Options** | Ad-Hoc Query Report in the CPRS GUI on the Reports tab | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  | ORWRP2 | ^DIC, ^DID, ^GMTSU, ^PSS50P7, ^TIULX, ^YTQGMTS |

| Routines | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** | None | | | | |
| **Related Protocols** | None | | | | |
| **Related Integration Control Registrations (ICRs)** | 2825 and 3148 | | | | |
| **Data Passing** | Input | Output Reference | Both | Global Reference | Local |
| **Input Attribute Name and Definition** | Name: GMTSRT  Definition: Root ^TMP global name where data is passed from the GUI.  Name: GMTSFI  Definition: FileMan file number that is being used for selection.  Name: GMTSFM  Definition: The entry in the file being searched where the list should begin (FROM).  Name: DIR  Definition: The ‘direction’ to search – either forward or backward | | | | |
| **Output Attribute Name and Definition** | Name: ^TMP(“ORDATA”,$J  Definition: Returns the listing of items to select from, based on the request from the GUI. | | | | |

| Current Logic |
| --- |
| Prior to these changes, GMTSADH5 didn’t allow selection from the PHARMACY ORDERABLE ITEM file (#50.7) or the VA DRUG CLASS file (#50.605) |
| Modified Logic (Changes are in bold) |
| GMTSADH5 ; SLC/DCM,KER - Health Summary Ad Hoc RPC's ;Nov 26, 2018@22:22  ;;2.7;Health Summary;\*\*36,35,37,49,63,110,116,125\*\*;Oct 20, 1995;Build  1  ;  ; External References  ; DBIA 1268 ^AUTTHF(  ; DBIA 1268 ^AUTTHF("B"  ; DBIA 67 ^LAB(60  ; DBIA 1256 ^PXD(811.9  ; DBIA 3059 ^TIU(8925.1  ; DBIA 10006 ^DIC  ; DBIA 2052 $$GET1^DID  ; DBIA 3058 $$ISA^TIULX  ; DBIA 6345 SEL^YTQGMTS  **; DBIA 4543 SSET^PSN50P65**  **; DBIA 4662 SSET^PSS50P7**  ;  COMP(Y) ; Get ADHOC sub components (FILE 142.1)  ;  ; Y(i)=(1)I;IFN^(2)Component Name [Abb]^(3)Occ Limit^  ; (4)Time Limit^(5)Header Name^(6)Hosp Loc Disp^  ; (7)ICD Text Disp^(8)Prov Narr Disp^  ; (9)CPT Modifier Disp^(10)Summary Order  ;  N GMTSI,GMTSII,GMTSIFN,GMTSC,X,X1  S Y(1)=$O(^GMT(142,"B","GMTS HS ADHOC OPTION",0))  I 'Y(1) S Y(1)=-1 Q ; Error, no ADHOC type definedet subcomponents from a predefined ADHOC component  ; GMTSUB=desired Adhoc subcomponent  ; Y(i)=ifn of pointed to file entry^name  Q:'$G(GMTSUB)  N GMTSI,GMTSII,GMTSIFN,GMTSC,X,X1  S X=$O(^GMT(142,"B","GMTS HS ADHOC OPTION",0))  I 'X Q ; Error, no ADHOC type defined  S (GMTSC,GMTSI)=0,GMTSII=X  F S GMTSI=$O(^GMT(142,GMTSII,1,GMTSUB,1,GMTSI)) Q:'GMTSI S X=^(GMTSI,  0) D  . S GMTSIFN=+X,X1=$P(X,";",2)  . I '$D(@("^"\_X1\_+X\_",0)")) Q  . S X=@("^"\_X1\_+X\_",0)"),GMTSC=GMTSC+1,Y(GMTSC)=GMTSIFN\_"^"\_$P(X,"^")  Q  ;  FILES(Y,GMTSCP) ; Get Files to select from for a component  Q:'$G(GMTSCP) Q:'$D(^GMT(142.1,GMTSCP,1))  N GMTSGEC,GMTSI,GMTSC,X  S (GMTSGEC,GMTSI,GMTSC)=0  I $P($G(^GMT(142.1,GMTSCP,0)),U,4)="GECH" S GMTSGEC=1  F S GMTSI=$O(^GMT(142.1,GMTSCP,1,GMTSI)) Q:'GMTSI D  .S X=^(GMTSI,0),GMTSC=GMTSC+1 S:GMTSGEC=1 X=X\_"G**" ;naked reference refe**  rs to ^GMT(142.1,GMTSCP,1,GMTSI from previous line  .S Y(GMTSC)=GMTSI\_"^"\_$$FNAM^GMTSU(+X)\_"^"\_X  Q  ;  FILESEL(GMTSRT,GMTSFI,GMTSFM,DIR) ; Get file entries  Q:'$G(GMTSFI)  K ^TMP("ORDATA",$J)  N GMTSI,GMTSJ,GMTSC,X,GMTSGL,GMTSGLB,GMTSCNT,HFC  S GMTSI=$G(GMTSFM),GMTSCNT=44,GMTSC=0,GMTSRT=$NA(^TMP("ORDATA",$J,1))  S:'$D(DIR) DIR=1  I GMTSFI=60 D Q  . F Q:GMTSC'<GMTSCNT S GMTSI=$O(^LAB(60,"B",GMTSI),DIR) Q:GMTSI="" S  GMTSJ=0 F S GMTSJ=$O(^LAB(60,"B",GMTSI,GMTSJ)) Q:'GMTSJ D  . . I $D(^LAB(60,GMTSJ,0)) S X=^(0) I $P(X,"^",4)="CH","BO"[$P(X,"^",3)  S GMTSC=GMTSC+1,^TMP("ORDATA",$J,1,GMTSC)=GMTSJ\_"^"\_GMTSI  I GMTSFI="9999999.64G" D Q  . F Q:GMTSC'<GMTSCNT S GMTSI=$O(^AUTTHF("B",GMTSI),DIR) Q:GMTSI="" S  GMTSJ=0 F S GMTSJ=$O(^AUTTHF("B",GMTSI,GMTSJ)) Q:'GMTSJ I $D(^AUTTHF(GMTSJ,0)  ) S X=^(0) D  **..;naked references below refers to ^AUTTHF(GMTSJ,0**  ..I (($P(^(0),U,10)="C")&(+$P(^(0),U,11)'=1))&($P(^(0)," ",1)="GEC") D  ...S GMTSC=GMTSC+1  ...S HFC=$S($P($G(X),U,10)="F":"Factor",$P($G(X),U,10)="C":"Category")  ...S ^TMP("ORDATA",$J,1,GMTSC)=GMTSJ\_U\_GMTSI\_" ("\_HFC\_")"  I GMTSFI=9999999.64 D Q  . F Q:GMTSC'<GMTSCNT S GMTSI=$O(^AUTTHF("B",GMTSI),DIR) Q:GMTSI="" S  GMTSJ=0 F S GMTSJ=$O(^AUTTHF("B",GMTSI,GMTSJ)) Q:'GMTSJ I $D(^AUTTHF(GMTSJ,0)  ) S X=^(0) D  ..I +$P(X,U,11)'=1 D  ...S GMTSC=GMTSC+1  ...S HFC=$S($P($G(X),U,10)="F":"Factor",$P($G(X),U,10)="C":"Category")  ...S ^TMP("ORDATA",$J,1,GMTSC)=GMTSJ\_U\_GMTSI\_" ("\_HFC\_")"  I GMTSFI=811.9 D Q  . F Q:GMTSC'<GMTSCNT S GMTSI=$O(^PXD(811.9,"B",GMTSI),DIR) Q:GMTSI=""  S GMTSJ=0 F S GMTSJ=$O(^PXD(811.9,"B",GMTSI,GMTSJ)) Q:'GMTSJ I $D(^PXD(811.9  ,GMTSJ,0)) S X=^(0) D  . . I $P(X,"^",6)'=1 S GMTSC=GMTSC+1,^TMP("ORDATA",$J,1,GMTSC)=GMTSJ\_"^  "\_GMTSI  I GMTSFI=8925.1 D Q  . F Q:GMTSC'<GMTSCNT S GMTSI=$O(^TIU(8925.1,"B",GMTSI),DIR) Q:GMTSI="  " S GMTSJ=0 F S GMTSJ=$O(^TIU(8925.1,"B",GMTSI,GMTSJ)) Q:'GMTSJ I $D(^TIU(892  5.1,GMTSJ,0)) S X=^(0) D  . . I $P(X,"^",4)="DOC",$$ISA^TIULX(GMTSJ,3) S GMTSC=GMTSC+1,^TMP("ORDA  TA",$J,1,GMTSC)=GMTSJ\_"^"\_GMTSI  ; KDM 1/28/2014 GMTS\*2.7\*110  ; Added code for the 81 file (CPT code) to list in logical sort ord  er using the "BA" indexed file  I GMTSFI=81 D Q  . S GMTSGL=$$FCLR^GMTSU(+GMTSFI) I $L(GMTSGL) S GMTSGLB=$$FLOC^GMTSU(+G  MTSFI)\_"""BA"")" D  .. F Q:GMTSC'<GMTSCNT S GMTSI=$O(@GMTSGLB@(GMTSI),DIR) Q:GMTSI="" S  GMTSJ=0 F S GMTSJ=$O(@GMTSGLB@(GMTSI,GMTSJ)) Q:'GMTSJ I $D(@GMTSGL@(GMTSJ,0))  S X=^(0) D  ... S GMTSC=GMTSC+1,^TMP("ORDATA",$J,1,GMTSC)=GMTSJ\_"^"\_GMTSI  ; use Mental Health API to return active, scoreable instruments  I GMTSFI=601.71 D Q  . N GMTSDIR S GMTSDIR=DIR  . D SEL^YTQGMTS(GMTSRT,GMTSI,GMTSCNT,GMTSDIR)  **I GMTSFI=50.7 D Q**  **. D SSET^PSS50P7(GMTSC,GMTSCNT,GMTSI,DIR,"ORDATA")**  **I GMTSFI=50.605 D Q**  **. D SSET^PSN50P65(GMTSC,GMTSCNT,GMTSI,DIR,"ORDATA")**  S GMTSGL=$$FCLR^GMTSU(+GMTSFI) I $L(GMTSGL) S GMTSGLB=$$FLOC^GMTSU(+GMT  SFI)\_"""B"")" D  . F Q:GMTSC'<GMTSCNT S GMTSI=$O(@GMTSGLB@(GMTSI),DIR) Q:GMTSI="" S G  MTSJ=0 F S GMTSJ=$O(@GMTSGLB@(GMTSI,GMTSJ)) Q:'GMTSJ I $D(@GMTSGL@(GMTSJ,0)) S  X=^(0) D  . . S GMTSC=GMTSC+1,^TMP("ORDATA",$J,1,GMTSC)=GMTSJ\_"^"\_GMTSI  Q  ;  REPORT(GMTSEG,GMTSEGC,GMTSEGI,GMTSCPS,DFN) ; Build Report  ; Uses array of Components passed in GMTSCPS()  ; GMTSCPS(i)=array of subcomponents chosen,  ; value is pointer at ^GMT(142,DA(1),1,DA)  Q:'$G(DFN)  N GMTSCNT,DIC,DIZ,DIW,DIWI,DIWT,DIWTC,X,GMTSI,GMTSJ,GMTSK,GMTSTYP,GMTST  ITL  S X="GMTS HS ADHOC",DIC=142,DIZ(0)="ZF"  D ^DIC Q:'Y  S GMTSTYP=+Y,GMTSTITL="AD HOC",(GMTSJ,GMTSI)=0,GMTSEGC=$O(GMTSCPS(99999  999),-1)  F S GMTSI=$O(GMTSCPS(GMTSI)) Q:'GMTSI D  . N GMTSREC,GMTSS2,GMTSSJ,GMTSEL  . S GMTSREC=^GMT(142,GMTSTYP,1,+GMTSCPS(GMTSI),0),GMTSJ=GMTSJ+1  . S GMTSEG(GMTSJ)=GMTSREC,GMTSEGI($P(GMTSREC,U,2))=GMTSJ,GMTSS2=0,GMTSS  J=GMTSJ  . S $P(GMTSEG(GMTSJ),"^",3)=$P(GMTSCPS(GMTSI),"^",2)  . S $P(GMTSEG(GMTSJ),"^",4)=$P(GMTSCPS(GMTSI),"^",3)  . I $L($P(GMTSCPS(GMTSI),"^",4)) S $P(GMTSEG(GMTSJ),"^",5)=$P(GMTSCPS(G  MTSI),"^",4)  . I $L($P(GMTSCPS(GMTSI),"^",5)) S $P(GMTSEG(GMTSJ),"^",6)=$P(GMTSCPS(G  MTSI),"^",5)  . S $P(GMTSEG(GMTSJ),"^",7)=$P(GMTSCPS(GMTSI),"^",6)  . I $L($P(GMTSCPS(GMTSI),"^",7)) S $P(GMTSEG(GMTSJ),"^",8)=$P(GMTSCPS(G  MTSI),"^",7)  . S (GMTSCNT,GMTSK)=0  . F S GMTSK=$O(GMTSCPS(GMTSK)) Q:'GMTSK D  . .I $P($G(GMTSCPS(GMTSK)),U,9)="9999999.64G" S $P(GMTSCPS(GMTSK),U,9)=  "9999999.64"  . .I +GMTSCPS(GMTSI)=+GMTSCPS(GMTSK),$P(GMTSCPS(GMTSK),"^",9),$P(GMTSCP  S(GMTSK),"^",10) D  . . . S GMTSCNT=GMTSCNT+1  . . . S:'$D(GMTSEG(GMTSJ,$P(GMTSCPS(GMTSK),"^",9),0)) GMTSEG(GMTSJ,$P(G  MTSCPS(GMTSK),"^",9),0)=$$GET1^DID($P(GMTSCPS(GMTSK),"^",9),,,"GLOBAL NAME")  . . . S GMTSEG(GMTSJ,$P(GMTSCPS(GMTSK),"^",9),GMTSCNT)=$P(GMTSCPS(GMTSK  ),"^",10)  . . . K GMTSCPS(GMTSK)  Q  ;  SUBITEM(Y,GMTSTEST) ; Get Subitems for a Test Panel  Q:'$G(GMTSTEST) N GMTSCNT S GMTSCNT=0  I '$L($P(^LAB(60,GMTSTEST,0),"^",5)),$O(^LAB(60,GMTSTEST,2,0)) D COMPIL  E(GMTSTEST,GMTSCNT)  Q  ;  COMPILE(GMTSTEST,GMTSCNT) ; Expand lab panels  N GMTSI,GMTSJ,GMTSRT S GMTSI=0  F S GMTSI=$O(^LAB(60,GMTSTEST,2,GMTSI)) Q:GMTSI'>0 D  . S GMTSJ=+$G(^LAB(60,GMTSTEST,2,+GMTSI,0))  . S GMTSRT=$G(^LAB(60,+GMTSJ,0))  . I $L($P(GMTSRT,U,5)),("BO"[$P(GMTSRT,U,3)) D  . . S GMTSCNT=GMTSCNT+1  . . S Y(GMTSCNT)=+GMTSJ\_"^"\_GMTSRT  . E D  . . D COMPILE(+$G(^LAB(60,GMTSTEST,2,GMTSI,0)),GMTSCNT)  Q |

Table 16 PSS50P7

| Routines | Activities | | | |
| --- | --- | --- | --- | --- |
| **Routine Name** | PSS50P7 | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** | 1041110, 1041111 | | | |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Routines | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output Reference | Both | Global Reference | Local |
| **Input Attribute Name and Definition** | Name:  Definition: | | | | |
| **Output Attribute Name and Definition** | Name:  Definition: | | | | |

| Current Logic |
| --- |
| Currently, there is no logic in this routine to return the required information. |
| Modified Logic (Changes are in bold) |
| PSS50P7 ;BIR/LDT - API FOR INFORMATION FROM FILE 50.7;Nov 15, 2018@16:43  ;;1.0;PHARMACY DATA MANAGEMENT;\*\*85,91,199,234\*\*;9/30/97;Build 1  ;  ZERO(PSSIEN,PSSFT,PSSFL,LIST) ;  ;PSSIEN - IEN of entry in PHARMACY ORDERABLE ITEM file (#50.7).  ;PSSFT - Free Text name in PHARMACY ORDERABLE ITEM file (#50.7).  ;PSSFL - Inactive flag - "" - All entries.  ; FileMan Date - Only entries with no Inactive Date or an Inacti  ve Date greater than this date.  ;LIST - Subscript of ^TMP array in the form ^TMP($J,LIST,Field Number w  here Field Number is the  ;Field Number of the data piece being returned.  ;Returns NAME field (#.01), DOSAGE FORM field (#.02), IV FLAG field (#.  03), INACTIVE DATE field (#.04),  ;DAY (nD) OR DOSE (nL) LIMIT field (#.05), MED ROUTE field (#.06), SCHE  DULE TYPE fiedl (#.07),  ;SCHEDULE field (#.08), SUPPLY field (#.09), FORMULARY STATUS field (#5  ), and NON-VA MED field (#8) of  ;PHARMACY ORDERABLE ITEM file (#50.7).  N DIERR,ZZERR,PSS50P7,SCR,PSS  I $G(LIST)']"" Q  K ^TMP($J,LIST)  I +$G(PSSIEN)'>0,($G(PSSFT)']"") S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUN  D" Q  I $G(PSSIEN)]"",+$G(PSSIEN)'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  S SCR("S")=""  I +$G(PSSFL)>0 N ND D SETSCRN  I +$G(PSSIEN)>0 N PSSIEN2 S PSSIEN2=$$FIND1^DIC(50.7,"","A","`"\_PSSIEN,  ,SCR("S"),"") D  .I +PSSIEN2'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND" Q  .S ^TMP($J,LIST,0)=1  .D GETS^DIQ(50.7,+PSSIEN2,".01;.02;.03;.04;.05;.06;.07;.08;.09;8;5","IE  ","PSS50P7") S PSS(1)=0  .F S PSS(1)=$O(PSS50P7(50.7,PSS(1))) Q:'PSS(1) D SETZRO^PSS50P7A  I +$G(PSSIEN)'>0,$G(PSSFT)]"" D  .I PSSFT["??" D LOOP^PSS50P7A(1) Q  .D FIND^DIC(50.7,,"@;.01;.02","QP",PSSFT,,"B",SCR("S"),,"")  .I +$G(^TMP("DILIST",$J,0))=0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  .I +^TMP("DILIST",$J,0)>0 S ^TMP($J,LIST,0)=+^TMP("DILIST",$J,0) N PSSX  X S PSSXX=0 F S PSSXX=$O(^TMP("DILIST",$J,PSSXX)) Q:'PSSXX D  ..S PSSIEN=+^TMP("DILIST",$J,PSSXX,0) K PSS50P7 D GETS^DIQ(50.7,+PSSIEN  ,".01;.02;.03;.04;.05;.06;.07;.08;.09;8;5","IE","PSS50P7") S PSS(1)=0  ..F S PSS(1)=$O(PSS50P7(50.7,PSS(1))) Q:'PSS(1) D SETZRO^PSS50P7A  K ^TMP("DILIST",$J)  Q  ;  SYNONYM(PSSIEN,PSSFT,PSSFL,LIST) ;  ;PSSIEN - IEN of entry in PHARMACY ORDERABLE ITEM file (#50.7).  ;PSSFT - Free Text name in PHARMACY ORDERABLE ITEM file (#50.7).  ;PSSFL - Inactive flag - 0 or "" - All entries.  ; FileMan Date - Only entries with no Inactive Date or an Inacti  ve Date greater than this date.  ;LIST - Subscript of ^TMP array in the form ^TMP($J,LIST,Field Number w  here Field Number is the  ;Field Number of the data piece being returned.  ;Returns NAME field (#.01), DOSAGE FORM field (#.02), SYNONYM subfile (  #50.72), and SYNONYM field (#.01)  ;of PHARMACY ORDERABLE ITEM file (#50.7).  N DIERR,ZZERR,PSS50P7,SCR,PSS,CNT  I $G(LIST)']"" Q  K ^TMP($J,LIST)  I +$G(PSSIEN)'>0,($G(PSSFT)']"") S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUN  D" Q  I $G(PSSIEN)]"",+$G(PSSIEN)'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  S SCR("S")=""  I +$G(PSSFL)>0 N ND D SETSCRN  I +$G(PSSIEN)>0 N PSSIEN2 S PSSIEN2=$$FIND1^DIC(50.7,"","A","`"\_PSSIEN,  ,SCR("S"),"") D  .I +PSSIEN2'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND" Q  .S ^TMP($J,LIST,0)=1  .D GETS^DIQ(50.7,+PSSIEN2,".01;.02;2\*","IE","PSS50P7") S PSS(1)=0  .S CNT=0 F S PSS(1)=$O(PSS50P7(50.72,PSS(1))) Q:'PSS(1) D SETSYN^PSS5  0P7A S CNT=CNT+1  .S ^TMP($J,LIST,+PSSIEN2,"SYN",0)=$S(CNT>0:CNT,1:"-1^NO DATA FOUND")  .S PSS(2)=0 F S PSS(2)=$O(PSS50P7(50.7,PSS(2))) Q:'PSS(2) D SETZR2^PS  S50P7A  I +$G(PSSIEN)'>0,$G(PSSFT)]"" D  .I PSSFT["??" D LOOP^PSS50P7A(2) Q  .D FIND^DIC(50.7,,"@;.01","QP",PSSFT,,"B",SCR("S"),,"")  .I +$G(^TMP("DILIST",$J,0))=0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  .I +^TMP("DILIST",$J,0)>0 S ^TMP($J,LIST,0)=+^TMP("DILIST",$J,0) N PSSX  X S PSSXX=0 F S PSSXX=$O(^TMP("DILIST",$J,PSSXX)) Q:'PSSXX D  ..S PSSIEN=+^TMP("DILIST",$J,PSSXX,0) K PSS50P7(50.72) K PSS50P7 D GETS  ^DIQ(50.7,+PSSIEN,".01;.02;2\*","IE","PSS50P7") S PSS(1)=0  ..S CNT=0 F S PSS(1)=$O(PSS50P7(50.72,PSS(1))) Q:'PSS(1) D SETSYN^PSS  50P7A S CNT=CNT+1  ..S ^TMP($J,LIST,+PSSIEN,"SYN",0)=$S(CNT>0:CNT,1:"-1^NO DATA FOUND")  ..S PSS(2)=0 F S PSS(2)=$O(PSS50P7(50.7,PSS(2))) Q:'PSS(2) D SETZR2^P  SS50P7A  K ^TMP("DILIST",$J)  Q  ;  NAME(PSSIEN) ;  ;PSSIEN - IEN of entry in PHARMACY ORDERABLE ITEM file (#50.7).  ;Returns NAME field (#.01) of PHARMACY ORDERABLE ITEM file (#50.7) and  DOSAGE FORM name in external form.  N DIERR,ZZERR,PSS50P7,PSS  I +$G(PSSIEN)'>0 Q ""  D GETS^DIQ(50.7,+PSSIEN,".01;.02","E","PSS50P7")  I '$D(PSS50P7) Q ""  Q $G(PSS50P7(50.7,PSSIEN\_",",.01,"E"))\_" "\_$G(PSS50P7(50.7,PSSIEN\_",",.  02,"E"))  ;  INSTR(PSSIEN,PSSFT,PSSFL,LIST) ;  ;PSSIEN - IEN of entry in PHARMACY ORDERABLE ITEM file (#50.7).  ;PSSFT - Free Text name in PHARMACY ORDERABLE ITEM file (#50.7).  ;PSSFL - Inactive flag - "" - All entries.  ; FileMan Date - Only entries with no Inactive Date or an Inacti  ve Date greater than this date.  ;LIST - Subscript of ^TMP array in the form ^TMP($J,LIST,Field Number w  here Field Number is the  ;Field Number of the data piece being returned.  ;Returns INS and INS1 nodes of PHARMACY ORDERABLE ITEM file (#50.7).  N DIERR,ZZERR,PSS50P7,SCR,PSS  I $G(LIST)']"" Q  K ^TMP($J,LIST)  I +$G(PSSIEN)'>0,($G(PSSFT)']"") S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUN  D" Q  I $G(PSSIEN)]"",+$G(PSSIEN)'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  S SCR("S")=""  I +$G(PSSFL)>0 N ND D SETSCRN  I +$G(PSSIEN)>0 N PSSIEN2 S PSSIEN2=$$FIND1^DIC(50.7,"","A","`"\_PSSIEN,  ,SCR("S"),"") D  .I +PSSIEN2'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND" Q  .S ^TMP($J,LIST,0)=1  .D GETS^DIQ(50.7,+PSSIEN2,".01;.02;7;7.1","IE","PSS50P7") S PSS(1)=0  .F S PSS(1)=$O(PSS50P7(50.7,PSS(1))) Q:'PSS(1) D SETPTI^PSS50P7A  I +$G(PSSIEN)'>0,$G(PSSFT)]"" D  .I PSSFT["??" D LOOP^PSS50P7A(3) Q  .D FIND^DIC(50.7,,"@;.01","QP",PSSFT,,"B",SCR("S"),,"")  .I +$G(^TMP("DILIST",$J,0))=0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  .I +^TMP("DILIST",$J,0)>0 S ^TMP($J,LIST,0)=+^TMP("DILIST",$J,0) N PSSX  X S PSSXX=0 F S PSSXX=$O(^TMP("DILIST",$J,PSSXX)) Q:'PSSXX D  ..S PSSIEN=+^TMP("DILIST",$J,PSSXX,0) K PSS50P7 D GETS^DIQ(50.7,+PSSIEN  ,".01;.02;7;7.1","IE","PSS50P7") S PSS(1)=0  ..F S PSS(1)=$O(PSS50P7(50.7,PSS(1))) Q:'PSS(1) D SETPTI^PSS50P7A  K ^TMP("DILIST",$J)  Q  ;  DRGIEN(PSSIEN,PSSFL,LIST) ;  ;PSSIEN - IEN of entry in PHARMACY ORDERABLE ITEM file (#50.7).  ;PSSFL - Inactive flag - "" - All entries.  ; FileMan Date - Only entries with no Inactive Date or an Inacti  ve Date greater than this date.  ;LIST - Subscript of ^TMP array in the form ^TMP($J,LIST,Field Number w  here Field Number is the  ;Field Number of the data piece being returned.  ;Returns entries from DRUG file (#50) linked to the Pharmacy Orderable  Item IEN passed, GENERIC NAME field (#.01),  ;DEA, SPECIAL HDLG field (#3), APPLICATION PACKAGES' USE field (#63), a  nd the INACTIVE DATE field (#100)  ;of DRUG file (#50).  N DIERR,ZZERR,PSS50P7,SCR,PSS  I $G(LIST)']"" Q  K ^TMP($J,LIST)  I +$G(PSSIEN)'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND" Q  S SCR("S")="I $D(^PS(50.7,""A50"",PSSIEN,Y))"  ;Naked reference below refers to ^PSDRUG(Y,"I")  I +$G(PSSFL)>0 S SCR("S")="S ND=$G(^(""I"")) I ((ND="""")&($D(^PS(50.7,  ""A50"",PSSIEN,Y))))!((ND>PSSFL)&($D(^PS(50.7,""A50"",PSSIEN,Y))))"  I +$G(PSSIEN)>0 D FIND^DIC(50,,"@;.01","QX",PSSIEN,,"ASP",SCR("S"),,"PS  S50P7")  I +PSS50P7("DILIST",0)'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND" Q  S ^TMP($J,LIST,0)=+PSS50P7("DILIST",0) S PSS(1)=0 D  .F S PSS(1)=$O(PSS50P7("DILIST",PSS(1))) Q:'PSS(1) S PSS(2)=0 F S PS  S(2)=$O(PSS50P7("DILIST",PSS(1),PSS(2))) Q:'PSS(2) S ^TMP($J,LIST,+PSS50P7("DIL  IST",PSS(1),PSS(2)))=""  Q  ;  IEN(PSSFT,PSSFL,LIST) ;  ;PSSFT - Free Text name in PHARMACY ORDERABLE ITEM file (#50.7).  ;PSSFL - Inactive flag - "" - All entries.  ; FileMan Date - Only entries with no Inactive Date or an Inacti  ve Date greater than this date.  ;LIST - Subscript of ^TMP array in the form ^TMP($J,LIST,Field Number w  here Field Number is the  ;Field Number of the data piece being returned.  ;Returns NAME field (#.01), and DOSAGE FORM field (#.02) of PHARMACY OR  DERABLE ITEM file (#50.7).  N DIERR,ZZERR,PSS50P7,SCR,PSS  I $G(LIST)']"" Q  K ^TMP($J,LIST)  I $G(PSSFT)']"" S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND" Q  S SCR("S")=""  I +$G(PSSFL)>0 D SETSCRN  I PSSFT["??" D LOOP^PSS50P7A(4) Q  I $G(PSSFT)]"" D FIND^DIC(50.7,,"@;.01;.02IE","QP",PSSFT,,"B",SCR("S"),  ,"")  I +$G(^TMP("DILIST",$J,0))'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  S ^TMP($J,LIST,0)=+^TMP("DILIST",$J,0) S PSS(2)=0 D  .F S PSS(2)=$O(^TMP("DILIST",$J,PSS(2))) Q:'PSS(2) D  ..S ^TMP($J,LIST,+^TMP("DILIST",$J,PSS(2),0),.01)=$P(^TMP("DILIST",$J,P  SS(2),0),"^",2)  ..S ^TMP($J,LIST,"B",$P(^TMP("DILIST",$J,PSS(2),0),"^",2),+^TMP("DILIST  ",$J,PSS(2),0))=""  ..S ^TMP($J,LIST,+^TMP("DILIST",$J,PSS(2),0),.02)=$S($P($G(^TMP("DILIST  ",$J,PSS(2),0)),"^",3)]"":$P(^TMP("DILIST",$J,PSS(2),0),"^",3,4),1:"")  K ^TMP("DILIST",$J)  Q  ;  LOOKUP(PSSFT,PSSD,PSSS,LIST) ;  ;PSSFT - Free Text name in PHARMACY ORDERABLE ITEM file (#50.7)  ;PSSD - Index being traversed.  ;PSSS - Screening information as defined in DIC("S").  ;LIST - Subscript of ^TMP array in the form ^TMP($J,LIST,Field Number w  here Field Number is the  ;Field Number of the data piece being returned.  ;Returns NAME field (#.01), DOSAGE FORM field (#.02), and INACTIVE DATE  field (#.04) of  ;PHARMACY ORDERABLE ITEM file (#50.7).  N DIERR,ZZERR,PSS50P7,SCR,PSS  I $G(LIST)']"" Q  K ^TMP($J,LIST)  I $G(PSSFT)']"" S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND" Q  D LOOKUP^PSS50P7A  Q  ;  SETSCRN ;Set Screen for inactive entries in PHARMACY ORDERABLE ITEM file (#50.7  ).  ;Naked reference below refers to ^PS(50.7,+Y,0)  S SCR("S")="S ND=$P($G(^(0)),U,4) I ND=""""!(ND>PSSFL)"  Q  **SSET(PSSC,PSSCNT,PSSI,DIR,SUB) ;Pull back a subset of the PHARMACY ORDERABLE ITE**  **M file (#50.7)**  **;**  **N PSSJ,X,Y**  **F Q:PSSC'<PSSCNT S PSSI=$O(^PS(50.7,"B",PSSI),DIR) Q:PSSI="" S PSSJ=**  **0 F S PSSJ=$O(^PS(50.7,"B",PSSI,PSSJ)) Q:'PSSJ I $D(^PS(50.7,PSSJ,0)) S X=^(0)**  **D**  **. S Y=$P($G(^PS(50.606,$P(X,"^",2),0)),"^"),PSSC=PSSC+1,^TMP(SUB,$J,1,P**  **SSC)=PSSJ\_"^"\_PSSI\_" "\_Y**  **Q** |

Table 17 PSN50P65

| Routines | Activities | | | |
| --- | --- | --- | --- | --- |
| **Routine Name** | PSN50P65 | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** | 1041112, 1041113, 1041114, 1041115, 1041116 | | | |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Routines | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output Reference | Both | Global Reference | Local |
| **Input Attribute Name and Definition** | Name:  Definition: | | | | |
| **Output Attribute Name and Definition** | Name:  Definition: | | | | |

| Current Logic |
| --- |
| Currently, there is no logic in this routine to return the required information. |
| Modified Logic (Changes are in bold) |
| PSN50P65 ;BIR/LDT - API FOR INFORMATION FROM FILE 50.605;Nov 16, 2018@10:30  ;;4.0;NATIONAL DRUG FILE;\*\*80,567\*\*;30 Oct 98;Build 1  ;  IEN(PSNIEN,PSNFT,LIST) ;  ;PSNIEN - IEN of entry in VA DRUG CLASS file (#50.605).  ;PSNFT - Free Text name in VA DRUG CLASS file (#50.605).  ;LIST - Subscript of ^TMP array in the form ^TMP($J,LIST,Field Number w  here Field Number is the  ; Field Number of the data piece beingreturned.  ;Returns CODE field (#.01), and CLASSIFICATION field (#1) of VA DRUG C  LASS file (#50.605).  N DIERR,ZZERR,PSN50P65,PSN  I $G(LIST)']"" Q  K ^TMP($J,LIST)  I +$G(PSNIEN)'>0,($G(PSNFT)']"") S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUN  D" Q  I $G(PSNIEN)]"",+$G(PSNIEN)'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  I $G(PSNIEN)]"" N PSNIEN2 S PSNIEN2=$$FIND1^DIC(50.605,"","B","`"\_PSNIE  N,,,"") D  .I +PSNIEN2'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND" Q  .S ^TMP($J,LIST,0)=1  .D GETS^DIQ(50.605,+PSNIEN2,".01;1","IE","PSN50P65") S PSN(1)=0  .F S PSN(1)=$O(PSN50P65(50.605,PSN(1))) Q:'PSN(1) D SETZRO  I $G(PSNIEN)="",$G(PSNFT)]"" D  .I PSNFT["??" D LOOP(1) Q  .D FIND^DIC(50.605,,"@;.01;1","QP",PSNFT,,"B",,,"")  .I +$G(^TMP("DILIST",$J,0))=0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  .I +^TMP("DILIST",$J,0)>0 S ^TMP($J,LIST,0)=+^TMP("DILIST",$J,0) N PSNX  X S PSNXX=0 F S PSNXX=$O(^TMP("DILIST",$J,PSNXX)) Q:'PSNXX D  ..S PSNIEN=+^TMP("DILIST",$J,PSNXX,0) K PSN50P65 D GETS^DIQ(50.605,+PSN  IEN,".01;1","IE","PSN50P65") S PSN(1)=0  ..F S PSN(1)=$O(PSN50P65(50.605,PSN(1))) Q:'PSN(1) D SETZRO  K ^TMP("DILIST",$J)  Q  ;  ROOT(PSNC) ;  ;Q "^PS(50.605, ""C"")" if PSNC is passed in as 1. If PSNC is null, Q  "^PS(50.605,"  I $G(PSNC)'=1 Q "^PS(50.605,"  Q "^PS(50.605,""C"")"  ;  C(PSNIEN,PSNFT,LIST) ;  ;PSNIEN - IEN of entry in VA DRUG CLASS file (#50.605).  ;PSNFT - Free Text name in VA DRUG CLASS file (#50.605).  ;LIST - Subscript of ^TMP array in the form ^TMP($J,LIST,Field Number w  here Field Number is the  ; Field Number of the data piece being returned.  ;Returns CODE field (#.01), CLASSIFICATION field (#1),PARENT CLASS fiel  d (#2), and TYPE field (#3)  ;of VA DRUG CLASS file (#50.605).  N DIERR,ZZERR,PSN50P65,PSN  I $G(LIST)']"" Q  K ^TMP($J,LIST)  I +$G(PSNIEN)'>0,($G(PSNFT)']"") S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUN  D" Q  I $G(PSNIEN)]"",+$G(PSNIEN)'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  I +$G(PSNIEN)>0 N PSNIEN2 S PSNIEN2=$$FIND1^DIC(50.605,"","A","`"\_PSNIE  N,"C",,"") D  .I +PSNIEN2'>0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND" Q  .S ^TMP($J,LIST,0)=1  .D GETS^DIQ(50.605,+PSNIEN2,".01;1;2;3","IE","PSN50P65") S PSN(1)=0  .F S PSN(1)=$O(PSN50P65(50.605,PSN(1))) Q:'PSN(1) D SETZRO2  I $G(PSNIEN)="",$G(PSNFT)]"" D  .I PSNFT["??" D LOOP(2) Q  .D FIND^DIC(50.605,,"@;.01;1","QP",PSNFT,,"C",,,"")  .I +$G(^TMP("DILIST",$J,0))=0 S ^TMP($J,LIST,0)=-1\_"^"\_"NO DATA FOUND"  Q  .I +^TMP("DILIST",$J,0)>0 S ^TMP($J,LIST,0)=+^TMP("DILIST",$J,0) N PSNX  X S PSNXX=0 F S PSNXX=$O(^TMP("DILIST",$J,PSNXX)) Q:'PSNXX D  ..S PSNIEN=+^TMP("DILIST",$J,PSNXX,0) K PSN50P65 D GETS^DIQ(50.605,+PSN  IEN,".01;1;2;3","IE","PSN50P65") S PSN(1)=0  ..F S PSN(1)=$O(PSN50P65(50.605,PSN(1))) Q:'PSN(1) D SETZRO2  K ^TMP("DILIST",$J)  Q  ;  SETZRO ;  S ^TMP($J,LIST,+PSN(1),.01)=$G(PSN50P65(50.605,PSN(1),.01,"I"))  S ^TMP($J,LIST,"B",$G(PSN50P65(50.605,PSN(1),.01,"I")),+PSN(1))=""  S ^TMP($J,LIST,+PSN(1),1)=$G(PSN50P65(50.605,PSN(1),1,"I"))  Q  ;  SETZRO2 ;  S ^TMP($J,LIST,+PSN(1),.01)=$G(PSN50P65(50.605,PSN(1),.01,"I"))  S ^TMP($J,LIST,"C",$G(PSN50P65(50.605,PSN(1),.01,"I")),+PSN(1))=""  S ^TMP($J,LIST,+PSN(1),1)=$G(PSN50P65(50.605,PSN(1),1,"I"))  S ^TMP($J,LIST,+PSN(1),2)=$S($G(PSN50P65(50.605,PSN(1),2,"I"))="":"",1:  PSN50P65(50.605,PSN(1),2,"I")\_"^"\_PSN50P65(50.605,PSN(1),2,"E"))  S ^TMP($J,LIST,+PSN(1),3)=$S($G(PSN50P65(50.605,PSN(1),3,"I"))="":"",1:  PSN50P65(50.605,PSN(1),3,"I")\_"^"\_PSN50P65(50.605,PSN(1),3,"E"))  Q  ;  LOOP(PSN) ;  N PSNIEN,CNT S CNT=0  S PSNIEN=0 F S PSNIEN=$O(^PS(50.605,PSNIEN)) Q:'PSNIEN D  .K PSN50P65 D GETS^DIQ(50.605,+PSNIEN,".01;1;2;3","IE","PSN50P65") S PS  N(1)=0 D  ..F S PSN(1)=$O(PSN50P65(50.605,PSN(1))) Q:'PSN(1) D @(PSN) S CNT=CN  T+1  S ^TMP($J,LIST,0)=$S(+CNT>0:CNT,1:"-1^NO DATA FOUND")  Q  1 ;  D SETZRO  Q  ;  2 ;  D SETZRO2  Q  **SSET(PSNC,PSNCNT,PSNI,DIR,SUB) ;Pull back a subset of the PHARMACY ORDERABLE ITE**  **M file (#50.7)**  **;**  **N PSNJ,X,Y**  **F Q:PSNC'<PSNCNT S PSNI=$O(^PS(50.605,"B",PSNI),DIR) Q:PSNI="" S PSN**  **J=0 F S PSNJ=$O(^PS(50.605,"B",PSNI,PSNJ)) Q:'PSNJ D**  **. S Y=$G(^PS(50.605,PSNJ,0)) I $P(Y,"^",2)["INACTIVE" Q**  **. I $P(Y,"^",3)="",PSNI'="HA000" Q**  **. S PSNC=PSNC+1,^TMP(SUB,$J,1,PSNC)=PSNJ\_"^"\_PSNI\_" - "\_$P(Y,"^",2)**  **Q** |

##### NSR 20141108: Require Patient Weight Method Entry

The release for this NSR is in patch GMRV\*5.0\*40

###### Mumps

Summary of changes

Routines

Table 15: Routine

| Routines | Activities | | | |
| --- | --- | --- | --- | --- |
| **Routine Name** | GMV40PST | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** |  | | | |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Routines | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output Reference | Both | Global Reference | Local |
| **Input Attribute Name and Definition** | Name:  Definition: | | | | |
| **Output Attribute Name and Definition** | Name:  Definition: | | | | |

| Current Logic |
| --- |
| This is a new routine |

| Modified Logic (Changes are in bold) |
| --- |
| GMV40PST ;HIOFO/FT - FIX PARAMETER SETTINGS ;Nov 21, 2018@11:15  ;;5.0;GEN. MED. REC. - VITALS;\*\*40\*\*;Oct 31, 2002;Build 4  ;;Per VA Directive 6402, this routine should not be modified.  Q  ; This routine uses the following IAs:  ; #10141 - MES^XPDUTL Kernel (supported)  ; #2263 - ^XPAR (Supported)  ;  EN ; main entry point  D XPAR  Q  XPAR ; Update the GUI version parameters. This subroutine is called during the  ; KIDS installation process.  ;  ; Variables:  ; GMV: [Private] Scratch  ; GMVGUI: [Private] Current version of GUI being installed  ; GMVLST: [Private] Scratch List  ;  ; NEW private variables  N GMV,GMVGUI,GMVMGUI,GMVDLL,GMVLST  ; Announce my intentions  D BMES^XPDUTL("Updating system parameters.")  ; Set current client version  S GMVGUI="5.0.40.1"  S GMVMGUI="5.0.40.1"  S GMVDLL="5.0.40.1"  ; Deactivate all previous versions of the Standalone and Manager from XPAR  D GETLST^XPAR(.GMVLST,"SYS","GMV GUI VERSION")  F GMV=0:0 S GMV=$O(GMVLST(GMV)) Q:'GMV D  . D EN^XPAR("SYS","GMV GUI VERSION",$P(GMVLST(GMV),"^",1),0)  . Q  ; Deactivate all previous versions of the DLL from XPAR  D GETLST^XPAR(.GMVLST,"SYS","GMV DLL VERSION")  F GMV=0:0 S GMV=$O(GMVLST(GMV)) Q:'GMV D  . D EN^XPAR("SYS","GMV DLL VERSION",$P(GMVLST(GMV),"^",1),0)  . Q  ; Add and/or activate current client versions  D EN^XPAR("SYS","GMV GUI VERSION","VITALS.EXE:"\_GMVGUI,1)  D EN^XPAR("SYS","GMV GUI VERSION","VITALSMANAGER.EXE:"\_GMVMGUI,1)  D EN^XPAR("SYS","GMV DLL VERSION",GMVDLL,1)  Q |

Table 15: Routine

| Routines | Activities | | | |
| --- | --- | --- | --- | --- |
| **Routine Name** | GMVPAR | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** |  | | | |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Routines | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output Reference | Both | Global Reference | Local |
| **Input Attribute Name and Definition** | Name:  Definition: | | | | |
| **Output Attribute Name and Definition** | Name:  Definition: | | | | |

| Current Logic |
| --- |
| Used by the Vitals GUIs to read and write system paramaters in VistA |

| Modified Logic (Changes are in bold) |
| --- |
| GMVPAR ; HOIFO/DP - XPARameter RPC ;Nov 28, 2018@14:46  ;;5.0;GEN. MED. REC. - VITALS;\*\*3\*\*;Oct 31, 2002  ; Integration Agreements:  ; IA# 2263 [Supported] XPAR parameter call.  ; IA# 2541 [Supported] Call to XUPARAM.  ; IA# 10060 [Supported] FILE 200 fields  ; IA# 10090 [Supported] FILE 4 references  ;  ; This routine supports the following IAs:  ; #4367 - GMV PARAMETER RPC is called at RPC (private)  ;  ;DELLST; [Procedure] Delete list of parameters  ;D NDEL^XPAR(ENT,PAR,.ERR)  ;S:'$G(ERR) @RESULTS@(0)="1^All instances removed"  ;Q  ;  DELPAR ; [Procedure] Delete single parameter value  D DEL^XPAR(ENT,PAR,INST,.ERR)  S:'$G(ERR) @RESULTS@(0)="1^Instance deleted"  Q  ;  ENTVAL ; [Procedure] Return value of the entity  I ENT="SYS" S ENT=$$KSP^XUPARAM("WHERE")  E I ENT="DIV" S ENT=$$GET1^DIQ(4,DUZ(2)\_",",.01)  E I ENT="USR" S ENT=$$GET1^DIQ(200,DUZ\_",",.01)  E S ENT=$$GET1^DIQ(+$P(ENT,"(",2),+ENT\_",",.01)  S @RESULTS@(0)=ENT  Q  ;  ;GETHDR; [Procedure] Returns common header format  ;S X=$$FIND1^DIC(8989.51,,"QX",PAR)  ;I X S @RESULTS@(0)=X\_";8989.51^"\_PAR  ;E S @RESULTS@(0)="-1^No such parameter ["\_PAR\_"]"  ;Q  ;  GETLST ; [Procedure] Return all instances of a parameter  D GETLST^XPAR(.RET,ENT,PAR,"E",.ERR)  Q:$G(ERR,0)  S TMP="RET"  F S TMP=$Q(@TMP) Q:TMP="" D  .S @RESULTS@($O(@RESULTS@(""),-1)+1)=@TMP  S @RESULTS@(0)=$O(@RESULTS@(""),-1)  Q  ;  GETPAR ; [Procedure] Returns external value of a parameter  S @RESULTS@(0)=$$GET^XPAR(ENT,PAR,INST,"E")  Q  ;  ;GETWP; [Procedure] Returns WP text for a parameter  ;D GETWP^XPAR(.RET,ENT,PAR,INST,.ERR)  ;Q:$G(ERR,0)  ;S TMP="RET"  ;F S TMP=$Q(@TMP) Q:TMP="" D  ;.S @RESULTS@($O(@RESULTS@(""),-1)+1)=@TMP  ;S @RESULTS@(0)=$O(@RESULTS@(""),-1)\_U\_INST  ;Q  ;  RPC(RESULTS,OPTION,ENT,PAR,INST,VAL) ; [Procedure] Main RPC Hit Point  ; RPC: [GMV PARAMETER]  ;  ; Requires that the parameter name in PAR  ; be in the GMV namespace.  ;  ; Input parameters  ; 1. RESULTS [Literal/Required] No description  ; 2. OPTION [Literal/Required] No description  ; 3. ENT [Literal/Required] No description  ; 4. PAR [Literal/Required] No description  ; 5. INST [Literal/Required] No description  ; 6. VAL [Literal/Required] No description  ;  N ERR,TMP,RET,TXT,IEN,IENS,ROOT  S INST=$G(INST,1)  S PAR=$G(PAR,"GMV")  S RESULTS=$NA(^TMP($J)) K @RESULTS  I PAR'?1"GMV".E S ^TMP($J,0)="-1^Non Vitals Measurements Parameter" Q  D:$T(@OPTION)]"" @OPTION  I +$G(ERR) K @RESULTS S @RESULTS@(0)="-1^Error: "\_(+ERR)\_" "\_$P(ERR,U,2)  I '$D(^TMP($J)) S @RESULTS@(0)="-1^No date returned"  D CLEAN^DILF  Q  ;  ;SETLST ; [Procedure] Build list of parameters  ;N GMVINS ; Instance Counter  ;D DELLST(ENT,PAR)  ;S GMVINS=""  ;F S GMVINS=$O(VAL(GMVINS)) Q:GMVINS="" D  ;.D EN^XPAR(ENT,PAR,GMVINS,VAL(GMVINS),.ERR)  ;S:'$G(ERR) @RESULTS@(0)="1^List "\_PAR\_" rebuilt"  ;Q  ;  SETPAR ; [Procedure] Set single value into a parameter  D EN^XPAR(ENT,PAR,INST,VAL,.ERR)  S:'$G(ERR) @RESULTS@(0)="1^Parameter updated"  Q  ;  ;SETWP; [Procedure] Set WP text into a parameter  ;S TXT=INST,TMP=""  ;F S TMP=$O(VAL(TMP)) Q:TMP="" D  ;.S TXT($O(TXT(""),-1)+1,0)=VAL(TMP)  ;D EN^XPAR(ENT,PAR,INST,.TXT,.ERR)  ;S:'$G(ERR) @RESULTS@(0)="1^WP Text Saved"  ;Q  ;  **SETLST ; [Procedure] Set parameter from list**  **N GMVINS ; Instance Counter**  **S GMVINS=""**  **F S GMVINS=$O(VAL(GMVINS)) Q:GMVINS="" D**  **.D EN^XPAR(ENT,PAR,$P(VAL(GMVINS),U,1),$P(VAL(GMVINS),U,2),.ERR)**  **S:'$G(ERR) @RESULTS@(0)="1^List "\_PAR\_" updated"**  **;** |

Parameters

Table XX: Parameter

| Parameter | Activities | | | |
| --- | --- | --- | --- | --- |
| **Parameter Name** | GMV REQUIRES QUALIFIER | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** |  | | | |

| Current Logic |
| --- |
|  |

| Modified Logic (Changes are in bold) |
| --- |
| NAME: GMV REQUIRES QUALIFIER DISPLAY TEXT: Required qualifiers  MULTIPLE VALUED: Yes INSTANCE TERM: Qualifier IEN  VALUE TERM: Required VALUE DATA TYPE: free text  VALUE HELP: True or False if required of not  INSTANCE DATA TYPE: free text INSTANCE HELP: IEN for the vital  DESCRIPTION:  List of vital measurements and if they require qualifiers to be entered  or not  PRECEDENCE: 1 ENTITY FILE: SYSTEM |

Table XX: Parameter

| Parameter | Activities | | | |
| --- | --- | --- | --- | --- |
| **Parameter Name** | GMV EXCEPTION PURGE | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** |  | | | |

| Current Logic |
| --- |
|  |

| Modified Logic (Changes are in bold) |
| --- |
| NAME: GMV REQUIRES QUALIFIER DISPLAY TEXT: Required qualifiers  MULTIPLE VALUED: Yes INSTANCE TERM: Qualifier IEN  VALUE TERM: Required VALUE DATA TYPE: free text  VALUE HELP: True or False if required of not  INSTANCE DATA TYPE: free text INSTANCE HELP: IEN for the vital  DESCRIPTION:  List of vital measurements and if they require qualifiers to be entered  or not  PRECEDENCE: 1 ENTITY FILE: SYSTEM |

Table XX: Parameter

| Parameter | Activities | | | |
| --- | --- | --- | --- | --- |
| **Parameter Name** | GMV EXCEPTION LOGGER | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** |  | | | |

| Current Logic |
| --- |
|  |

| Modified Logic (Changes are in bold) |
| --- |
| NAME: GMRV EXCEPTION LOGGER  DISPLAY TEXT: Activate/Deactivate the Exception Logger  MULTIPLE VALUED: No VALUE TERM: Activated?  VALUE DATA TYPE: yes/no  DESCRIPTION:  When this parameter is set to "yes" the application will display a custom  access violation screen to the user as well as logging the error stack and  allowing this to be sent via an email (if OR CPRS EXCEPTION EMAIL is not  blank). The log files are currently stored in the following folder  Type <Enter> to continue or '^' to exit:  "C:\Users\USER'S DNS\AppData\Local\APPLICATION NAME\Unique Log File  Name.txt".  Example:  C:\Users\DNS\AppData\Local\CprsChart.exe\CPRS\_16136\_01\_13\_16\_16  \_43\_LOG.TXT  PRECEDENCE: 1 ENTITY FILE: SYSTEM |

Table XX: Parameter

| Parameter | Activities | | | |
| --- | --- | --- | --- | --- |
| **Parameter Name** | GMV EXCEPTION EMAIL | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** |  | | | |

| Current Logic |
| --- |
|  |

| Modified Logic (Changes are in bold) |
| --- |
| NAME: GMRV EXCEPTION EMAIL DISPLAY TEXT: Exception email address  MULTIPLE VALUED: Yes VALUE TERM: Email Address  VALUE DATA TYPE: free text  VALUE HELP: Email address used to pre populate error email  INSTANCE DATA TYPE: numeric  DESCRIPTION:  When the Exception Logger is enabled (OR CPRS EXCEPTION LOGGER) the user  has the ability to pre populate an email through Microsoft Outlook. If  this parameter is not empty than the user can email the error log and this  email address will be used for the pre population of that email.  PRECEDENCE: 1 ENTITY FILE: SYSTEM |

Table XX: Parameter

| Parameter | Activities | | | |
| --- | --- | --- | --- | --- |
| **Parameter Name** | GMV HELP DESK TEXT | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** |  | | | |

| Current Logic |
| --- |
|  |

| Modified Logic (Changes are in bold) |
| --- |
| NAME: GMRV HELP DESK TEXT DISPLAY TEXT: Help text used by Vitals  MULTIPLE VALUED: No VALUE DATA TYPE: free text  VALUE HELP: Message to display to the user  DESCRIPTION:  This is the message that will inform the user on who to contact in case  of issues with Vitals.  Example: "your local CPRS help desk" or "your local IRM".  This will be used in sentences such as "Please contact your local help  desk for support"  PRECEDENCE: 1 ENTITY FILE: SYSTEM |

###### GUI

Vitals

Table 36: GUI

| Unit Name | Description |
| --- | --- |
| Vitals | The Vitals module is used to enter patient data, and is assigned to clinical staff |

fGMV\_UserMain

Summary of changes

Added new event for the file menu click to check to see if the user can edit templates or not and set the edit template menu item accordingly.

Detailed Changes

Table 40: Events

| Name | Type | Description |
| --- | --- | --- |
| mnFile | **Onclick** | **Perform RPC check to see if templates can be edited.** |

Table 46: Forms

| Forms | Description | | | |
| --- | --- | --- | --- | --- |
| **Form Name** | fGMV\_UserMain | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Form Functionality** | Main form for the Vitals standalone application | | | |

| Current Form Layout |
| --- |
|  |

| Modified Form Layout (Changes are in bold) |
| --- |
|  |

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | mnFileClick | | | |
| **Short Description** | Sets the Edit User Templates menu option’s enabled property based on if the user is able to edit or not. | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  | GMVAllowUserTemplates |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: The object that is calling into this procedure | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_UserMain.mnFileClick(Sender: TObject);  begin  mnFileEditUserTemplates.Enabled := GMVAllowUserTemplates;  end; |

Vitals Manager

Table 36: GUI

| Unit Name | Description |
| --- | --- |
| Vitals Manager | The Vitals Manager module is used to manage the Vitals templates and abnormal values ranges, and is assigned to the Clinical Application Coordinator, package coordinator, and Information Resource Management Service (IRMS) staff. |

fGMV\_Manager

Summary of changes

Added logic for the system wide requirments settings for the Vitals Measurments.

Detailed Changes

Table 46: Forms

| Forms | Description | | | |
| --- | --- | --- | --- | --- |
| **Form Name** | fGMV\_Manager | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Form Functionality** | Main form for the Vitals manager application | | | |

| Current Form Layout |
| --- |
|  |

| Modified Form Layout (Changes are in bold) |
| --- |
|  |

Uses Clause

Added uGMV\_Const to first uses section. Added AVCatcher, StrUtils to second uses section. Removed uGMV\_Const from this section

Table 39: Components on Form

| Name | Type | Description |
| --- | --- | --- |
| clbQualReq | TCheckListBox | List of Checkboxs used to set system wide requirement for specific qualifiers. |
| Label6 | TLabel | Label for clbQualReq |
| Removed Panel8 | TPanel | Was used for spacing and is no longer needed |
| PnlRequired | TPanel | Panel used to show requirments selection for Vitals Measurments |
| pnlReq | TPanel | Used as a header for the requirment section |
| pnlReqBody | TPanel | Panel that holds all the requirement options |
| chkQualReq | TCheckBox | Checkbox to indicate that the Vital Measurement is required (system wide) |
| tbtnSaveQualifier | TToolButton | Toolbutton used to save the system wide requirements (calls actQualifierSave) |
| actQualifierSave | TAction | Action to process the save of required fields |
| Savequalifier1 | TMenuItem | Menu item used to save the system wide requirements (calls actQualifierSave) |
| Removed Qualifiers1 | TMenuItem | Was an extra menu option no longer needed |
| lblReqQual | TLabel | Label used to describe the required qualifiers check list box |

clbQualReqClick

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | clbQualReqClick | | | |
| **Short Description** | This is a list of qualifiers for a specific Vitals Measurment. From here the user can set individual qualifers required when entering vitals data. | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Manager.clbQualReqClick(Sender: TObject);  var  aStrVal: String;  begin  aStrVal := TGMV\_FileEntry(tv.Selected.Data).IEN+'\_'+TGMV\_FileEntry(clbQualReq.Items.Objects[clbQualReq.ItemIndex]).IEN;  actQualifierSave.Enabled := IsQualifierGlobalRequired(aStrVal) <> clbQualReq.Checked[clbQualReq.ItemIndex];  end; |

chkQualReqClick

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | chkQualReqClick | | | |
| **Short Description** | This is used to set the Vitals Measurement as system wide qualifier required. With this set at least one qualifier will be required unless individually set via clbQualReq | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Manager.chkQualReqClick(Sender: TObject);  begin  actQualifierSave.Enabled := IsQualifierGlobalRequired(TGMV\_FileEntry(tv.Selected.Data).IEN) <> chkQualReq.Checked;  clbQualReq.Enabled := chkQualReq.Checked;  end; |

CMTemplateUpdated

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | CMTemplateUpdated | | | |
| **Short Description** | Windows message used to allow for the save template action to be available | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** |  | | | | |
| Definition: TMessage | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Manager.CMTemplateUpdated(var Message: TMessage);  begin  actFileSaveTemplate.Enabled := True;  end; |

CMTemplateRefreshed

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | CMTemplateRefreshed | | | |
| **Short Description** | Windows message used to allow for the save template action to be not available | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Message | | | | |
| Definition: TMessage | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Manager.CMTemplateRefreshed(var Message: TMessage);  begin  actFileSaveTemplate.Enabled := False;  end; |

tvChange

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | tvChange | | | |
| **Short Description** | Change event that is fired when switching between Vitals Measurements. We need to ensure that we load up all information for the specific measurement. | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  | Onchange event |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject  Name: Node  Definition: TTreeNode | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| procedure TfrmGMV\_Manager.tvChange(Sender: TObject; Node: TTreeNode);  var  SL: TStringList;  li:TListItem;  i, j: integer;  begin  pgctrl.ActivePage := tbshtBlank;  actFileSaveTemplate.Enabled := False;  actFileDeleteTemplate.Enabled := False;  actFileMakeDefault.Enabled := False;  actAbnormalSave.Enabled := False;  actSaveParameters.Enabled := False;  fraGMV\_EditTemplate1.acUp.Enabled := False;  fraGMV\_EditTemplate1.acDown.Enabled := False;  if Node.Data <> nil then  begin  if (TObject(Node.Data) is TGMV\_Template) then  begin  fraGMV\_EditTemplate1.acUp.Enabled := True;  fraGMV\_EditTemplate1.acDown.Enabled := True;  pgctrl.ActivePage := tbshtTemplate;  fraGMV\_EditTemplate1.EditTemplate := TGMV\_Template(Node.Data);  end  else if (TObject(Node.Data) is TGMV\_FileEntry) then  begin  for i := 0 to clbxQualifiers.Items.Count - 1 do  clbxQualifiers.Checked[i] := False;  clbxQualifiers.ItemIndex := -1;  clbxQualifiers.Enabled := False;  lvCategories.Items.Clear;//AAN 07/15/2002  SL := getCategoryQualifiers(TGMV\_FileEntry(Node.Data).IEN);  for i := 1 to SL.Count - 1 do  begin  j := GMVCats.IndexOfIEN(Piece(SL[i], '^', 1));  if j > - 1 then  begin  li := lvCategories.Items.Add;  li.Caption := {TitleCase}(GMVCats.Entries[j]);  li.SubItems.Add(TGMV\_FileEntry(GMVCats.Entries.Objects[j]).IEN);  li.SubItems.Add('[' + {TitleCase}(Piece(SL[i], '^', 3)) + ']');  end;  end;  SL.Free;  pgctrl.ActivePage := tbshtVitals;  end  else if (TObject(Node.Data) is TGMV\_VitalHiLoDefinition) then  with TGMV\_VitalHiLoDefinition(Node.Data) do  begin  case VitalType of  hltO2Sat:  fraLowValue.SetUpFrame(VitalType, hlLow, Minimum, Maximum, Increment)  else  begin  fraLowValue.SetUpFrame(VitalType, hlLow, Minimum, Maximum, Increment);  fraHighValue.SetUpFrame(VitalType, hlHigh, Minimum, Maximum, Increment);  end;  end;  fraHighValue.Visible := (VitalType <> hltO2Sat);  pgctrl.ActivePage := tbshtVitalsHiLo;  end  else  pgctrl.ActivePage := tbshtBlank;  actFileSaveTemplate.Enabled := (TObject(Node.Data) is TGMV\_Template);  actFileDeleteTemplate.Enabled := (TObject(Node.Data) is TGMV\_Template);  actFileMakeDefault.Enabled := (TObject(Node.Data) is TGMV\_Template);  actAbnormalSave.Enabled := (TObject(Node.Data) is TGMV\_VitalHiLoDefinition);  end  else if Node = FParamRoot then  begin  fraSystemParameters.LoadParameters;  actSaveParameters.Enabled := True;  pgctrl.ActivePage := tbshtSystemParameters;  end;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Manager.tvChange(Sender: TObject; Node: TTreeNode);  var  SL: TStringList;  li:TListItem;  i, j: integer;  begin  pgctrl.ActivePage := tbshtBlank;  actFileSaveTemplate.Enabled := False;  actFileDeleteTemplate.Enabled := False;  actFileMakeDefault.Enabled := False;  actAbnormalSave.Enabled := False;  actSaveParameters.Enabled := False;  fraGMV\_EditTemplate1.acUp.Enabled := False;  fraGMV\_EditTemplate1.acDown.Enabled := False;  if Node.Data <> nil then  begin  if (TObject(Node.Data) is TGMV\_Template) then  begin  fraGMV\_EditTemplate1.acUp.Enabled := True;  fraGMV\_EditTemplate1.acDown.Enabled := True;  pgctrl.ActivePage := tbshtTemplate;  fraGMV\_EditTemplate1.EditTemplate := TGMV\_Template(Node.Data);  end  else if (TObject(Node.Data) is TGMV\_FileEntry) then  begin  for i := 0 to clbxQualifiers.Items.Count - 1 do  clbxQualifiers.Checked[i] := False;  clbxQualifiers.ItemIndex := -1;  clbxQualifiers.Enabled := False;  **//set the required checkbox**  **chkQualReq.Checked := IsQualifierGlobalRequired(TGMV\_FileEntry(Node.Data).IEN);**  lvCategories.Items.Clear;//AAN 07/15/2002  **clbQualReq.Items.Clear;**  SL := getCategoryQualifiers(TGMV\_FileEntry(Node.Data).IEN);  **try**  for i := 1 to SL.Count - 1 do  begin  j := GMVCats.IndexOfIEN(Piece(SL[i], '^', 1));  if j > - 1 then  begin  li := lvCategories.Items.Add;  li.Caption := {TitleCase}(GMVCats.Entries[j]);  li.SubItems.Add(TGMV\_FileEntry(GMVCats.Entries.Objects[j]).IEN);  li.SubItems.Add('[' + {TitleCase}(Piece(SL[i], '^', 3)) + ']');  **//Fill out required categories**  **clbQualReq.Items.AddObject(GMVCats.Entries[j], TGMV\_FileEntry(GMVCats.Entries.Objects[j]));**  **clbQualReq.Checked[(clbQualReq.Count - 1)] := IsQualifierGlobalRequired(TGMV\_FileEntry(Node.Data).IEN+'\_'+TGMV\_FileEntry(GMVCats.Entries.Objects[j]).IEN);**  end;  end;  **finally**  SL.Free;  **end;**  **clbQualReq.Enabled := chkQualReq.Checked;**  pgctrl.ActivePage := tbshtVitals;  end  else if (TObject(Node.Data) is TGMV\_VitalHiLoDefinition) then  with TGMV\_VitalHiLoDefinition(Node.Data) do  begin  case VitalType of  hltO2Sat:  fraLowValue.SetUpFrame(VitalType, hlLow, Minimum, Maximum, Increment)  else  begin  fraLowValue.SetUpFrame(VitalType, hlLow, Minimum, Maximum, Increment);  fraHighValue.SetUpFrame(VitalType, hlHigh, Minimum, Maximum, Increment);  end;  end;  fraHighValue.Visible := (VitalType <> hltO2Sat);  pgctrl.ActivePage := tbshtVitalsHiLo;  end  else  pgctrl.ActivePage := tbshtBlank;  actFileSaveTemplate.Enabled := (TObject(Node.Data) is TGMV\_Template);  actFileDeleteTemplate.Enabled := (TObject(Node.Data) is TGMV\_Template);  actFileMakeDefault.Enabled := (TObject(Node.Data) is TGMV\_Template);  actAbnormalSave.Enabled := (TObject(Node.Data) is TGMV\_VitalHiLoDefinition);  end  else if Node = FParamRoot then  begin  fraSystemParameters.LoadParameters;  actSaveParameters.Enabled := True;  pgctrl.ActivePage := tbshtSystemParameters;  end;  end; |

actQualifierSaveExecute

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | actQualifierSaveExecute | | | |
| **Short Description** | Process the save of the requirement settings for the specific Vitals Measurement. | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Manager.actQualifierSaveExecute(Sender: TObject);  var  aRtnStr, aHelpTxt, aStrVal: String;  aParamLst: TStringList;  I: integer;  begin  //save the global required  aParamLst := TStringList.Create;  try  aParamLst.NameValueSeparator := '^';  aParamLst.Values[TGMV\_FileEntry(tv.Selected.Data).IEN] := IfThen(chkQualReq.Checked, 'True', 'False');  for I := 0 to clbQualReq.Count - 1 do  begin  aStrVal := TGMV\_FileEntry(tv.Selected.Data).IEN+'\_'+TGMV\_FileEntry(clbQualReq.Items.Objects[i]).IEN;  aParamLst.Values[aStrVal] := IfThen(clbQualReq.Checked[I], 'True', 'False');  end;  aRtnStr := setSystemParameterList(GMV\_QualifierRequired, '', aParamLst);  if Piece(aRtnStr, '^', 1) <> '1' then  begin  aHelpTxt := getSystemParameterByName('GMRV HELP DESK TEXT');  MessageDlg('Error marking qualifier required. Please contact ' + aHelpTxt + ' for help', mtError, [mbOK], 0);  end;  for I := 0 to aParamLst.Count - 1 do  begin  UpdateQualifierGlobalRequired(aParamLst.Names[i], aParamLst.ValueFromIndex[i]);  actQualifierSave.Enabled := false;  end;  finally  aParamLst.Free;  end;  end; |

LoadExceptionLogger

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | LoadExceptionLogger | | | |
| **Short Description** | Initializes the access violation logger. | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| Procedure TfrmGMV\_Manager.LoadExceptionLogger;  var  TmpLst: TStringList;  TmpStr: String;  begin  ExceptionLog.DaysToPurge := StrToIntDef(getSystemParameterByName('GMV EXCEPTION PURGE'), 60);  ExceptionLog.Enabled := Uppercase(getSystemParameterByName('GMV EXCEPTION LOGGER')) = 'YES';  TmpLst := getSystemParameterListByName('GMV EXCEPTION EMAIL');  try  for TmpStr in TmpLst do  ExceptionLog.EmailTo.Add(Piece(TmpStr, '^', 2));  finally  TmpLst.Free;  end;  end; |

RestoreSettings

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | RestoreSettings | | | |
| **Short Description** | Used to initialize the Vitals Manager and the Vitals Measurements. | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| procedure TfrmGMV\_Manager.RestoreSettings;  var  i: integer;  begin  Font := Screen.MenuFont;  for i := 0 to pgctrl.PageCount - 1 do  TTabSheet(pgctrl.Pages[i]).TabVisible := False;  pgctrl.ActivePage := tbshtBlank;  tv.Enabled := False;  tv.Items.Clear;  tv.ShowRoot := False;  with tv.Items.Add(nil, 'Loading, please wait...') do  begin  ImageIndex := -1;  selectedIndex := -1;  Selected := False;  end;  Show;  Application.ProcessMessages;  LoadTreeView;  tv.Enabled := True;  for i := 0 to GMVQuals.Entries.Count - 1 do  //AAN 07/18/2002: We should show Qualifier names as they are kept in DB  clbxQualifiers.Items.AddObject({TitleCase}(GMVQuals.Entries[i]), GMVQuals.Entries.Objects[i]);  clbxQualifiers.Enabled := False;  Caption := 'Vitals Management. User: '+ GMVUser.Name + ' (' + GMVUser.Title +') Division: '+ GMVUser.Division;  lvCategoriesResize(nil);  sb.SimplePanel := true;  //{$IFDEF GMRV\*5\*8} //HDR  actQualifierNew.Enabled := False;  actQualifierEdit.Enabled := False;  //{$ENDIF}  end; |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Manager.RestoreSettings;  var  i: integer;  begin  Font := Screen.MenuFont;  for i := 0 to pgctrl.PageCount - 1 do  TTabSheet(pgctrl.Pages[i]).TabVisible := False;  pgctrl.ActivePage := tbshtBlank;  tv.Enabled := False;  tv.Items.Clear;  tv.ShowRoot := False;  with tv.Items.Add(nil, 'Loading, please wait...') do  begin  ImageIndex := -1;  selectedIndex := -1;  Selected := False;  end;  Show;  Application.ProcessMessages;  LoadTreeView;  tv.Enabled := True;  for i := 0 to GMVQuals.Entries.Count - 1 do  //AAN 07/18/2002: We should show Qualifier names as they are kept in DB  clbxQualifiers.Items.AddObject({TitleCase}(GMVQuals.Entries[i]), GMVQuals.Entries.Objects[i]);  clbxQualifiers.Enabled := False;  Caption := 'Vitals Management. User: '+ GMVUser.Name + ' (' + GMVUser.Title +') Division: '+ GMVUser.Division;  lvCategoriesResize(nil);  sb.SimplePanel := true;  //{$IFDEF GMRV\*5\*8} //HDR  actQualifierNew.Enabled := **True**;  actQualifierEdit.Enabled := False;  //{$ENDIF}  end; |

actFileSaveTemplateExecute

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | actFileSaveTemplateExecute | | | |
| **Short Description** | Saves the current edited template and sets the save action to disabled (until next modification) | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| procedure TfrmGMV\_Manager.actFileSaveTemplateExecute(Sender: TObject);  begin  fraGMV\_EditTemplate1.SaveTemplate;  tv.Selected.Text := TGMV\_Template(tv.Selected.Data).TemplateName;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Manager.actFileSaveTemplateExecute(Sender: TObject);  begin  fraGMV\_EditTemplate1.SaveTemplate;  tv.Selected.Text := TGMV\_Template(tv.Selected.Data).TemplateName;  **actFileSaveTemplate.Enabled := false;**  end; |

clbxQualifiersClickCheck

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | clbxQualifiersClickCheck | | | |
| **Short Description** | Currently disabled but fires off logic when adding qualifiers to a Vitals Measurement. New logic just reverts the checkbox since this is not allowed. | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Manager.clbxQualifiersClickCheck(Sender: TObject);  var  i: integer;  VitalIEN: string;  CategoryIEN: string;  QualifierIEN: string;  begin  {$IFDEF DISABLEQUALIFIERS}  **//Revert the checkbox**  **clbxQualifiers.Checked[clbxQualifiers.ItemIndex] := not clbxQualifiers.Checked[clbxQualifiers.ItemIndex];**  Exit;  {$ENDIF}  {$IFDEF GMV\*5\*8}  actQualifierEdit.Enabled := False;  {$ELSE}  actQualifierEdit.Enabled := True;  {$ENDIF}  VitalIEN := TGMV\_FileEntry(tv.Selected.Data).IEN;  CategoryIEN := lvCategories.ItemFocused.SubItems[0];  QualifierIEN := TGMV\_FileEntry(clbxQualifiers.Items.Objects[clbxQualifiers.ItemIndex]).IEN;  i := clbxQualifiers.ItemIndex;  if clbxQualifiers.Checked[i] then  addQualifier(VitalIEN,CategoryIEN,QualifierIEN)  else  delQualifier(VitalIEN,CategoryIEN,QualifierIEN);  lvCategories.ItemFocused.SubItems[1] := '[' + QualDisplayList + ']';  end; |

Vitals Common

frmGMV\_EditUserTemplates

Summary of changes

Cleaned up form since using alignments options as well as added requirement checkboxes (inherited)

Detailed Changes

Table 46: Forms

| Forms | Description | | | |
| --- | --- | --- | --- | --- |
| **Form Name** | frmGMV\_EditUserTemplates | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Form Functionality** |  | | | |

| Current Form Layout |
| --- |
|  |

| Modified Form Layout (Changes are in bold) |
| --- |
|  |

Vitals Data Entry

frmGMV\_Qualifiers

Summary of changes

Form used to fill out qualifiers for a Vitals Measuremenet..

Detailed Changes

Uses Clause

Added uGMV\_Template to first uses section. Added uGMV\_Utils into the second uses statement

Table 40: Events

| Name | Type | Description |
| --- | --- | --- |
| Form | **OnCloseQuery** |  |

Table 46: Forms

| Forms | Description | | | |
| --- | --- | --- | --- | --- |
| **Form Name** | frmGMV\_Qualifiers | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Form Functionality** |  | | | |

| Current Form Layout |
| --- |
|  |

| Modified Form Layout (Changes are in bold) |
| --- |
|  |

FormCloseQuery

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | FormCloseQuery | | | |
| **Short Description** | Performs check to ensure that required fields are filled out | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Qualifiers.FormCloseQuery(Sender: TObject;  var CanClose: Boolean);  Const  aMsg = '%s required';  Var  I: integer;  LookupIEN: String;  aFmtMsg: TStringList;  aMsgStr: String;  begin  CanClose := True;  if ModalResult = mrOk then  begin  aFmtMsg := TStringlist.Create;  try  for i := 0 to FPanelList.Count - 1 do  begin  LookupIEN := fGMV\_Template.IEN + '\_' + TGMV\_TemplateQualifierBox(FPanelList[i]).CategoryIEN;  if fGMV\_Template.IsRequired[LookupIEN] then  begin  if TGMV\_TemplateQualifierBox(FPanelList[i]).DefaultQualifierIEN = '' then  aFmtMsg.Add(TitleCase(GMVCats.Entries[GMVCats.IndexOfIEN(TGMV\_TemplateQualifierBox(FPanelList[i]).CategoryIEN)]) );  end;  end;  if aFmtMsg.Count > 0 then  begin  for I := 0 to aFmtMsg.Count - 1 do  begin  if aFmtMsg.Count > 1 then  begin  If(I = aFmtMsg.Count - 1) then  aMsgStr := Trim(aMsgStr) + ' and '  else if I > 0 then  aMsgStr := Trim(aMsgStr) + ', ';  end;  aMsgStr := Trim(aMsgStr) + ' ' + aFmtMsg.Strings[i] + ' ';  end;  if aFmtMsg.Count = 1 then  aMsgStr := Trim(aMsgStr) + ' is'  else  aMsgStr := Trim(aMsgStr) + ' are';  ShowMessage(Format(aMsg, [aMsgStr]));  CanClose := False;  end;  finally  aFmtMsg.Free;  end;  end;  end; |

SelectQualifiers

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | SelectQualifiers | | | |
| **Short Description** | Opens up the qualifers for a given Vitals Measurement and preselects the defaults and or current selections | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: VType | | | | |
| Definition: TVitalType  Name: Quals  Definition: string  Name: QualsDisplay  Definition: string  Name: Ctrl  Definition: TControl  Name: aValue  Definition: string | | | | |
| **Output Attribute Name and Definition** | Name: Result | | | | |
| Definition: Boolean | | | | |

| Current Logic |
| --- |
| function SelectQualifiers(VType: TVitalType; var Quals, QualsDisplay: string; Ctrl: TControl;aValue:String): Boolean;  var  s: String;//AAN 07/11/02 for Debugging only  \_QForm: TfrmGMV\_Qualifiers;  TQB,  QPanel: TGMV\_TemplateQualifierBox;  iOrder,  i, j: integer;  ii: Integer;  pt: TPoint;  RetList: TStrings;  begin  Result := False;  \_QForm := TfrmGMV\_Qualifiers.Create(Application);  with \_QForm do  try  pt := Ctrl.Parent.ClientToScreen(Point(Ctrl.Left, Ctrl.Top));  Left := pt.x;  Top := pt.y + Ctrl.Height;  RetList := getVitalQualifierList(GMVVitalTypeAbbv[VType]);  \_QForm.pnlTitle.Caption :=  piece(RetList.Text,'^',3) + ' Qualifiers';  for i := 0 to RetList.Count - 1 do  begin  // S := RetList[i];  if Piece(RetList[i], '^', 1) = 'C' then  begin  QPanel := TGMV\_TemplateQualifierBox.CreateParented(  // \_QForm.pnlMain,  \_QForm,\_QForm.sb,  GMVVitalTypeIEN[VType], piece(RetList[i], '^', 2),'');  FPanelList.Add(QPanel);  // \_QForm.Width := FPanelList.Count \* 100;  end;  end;  RetList.Free;  // if FPanelList.Count < 2 then  // \_QForm.Width := 150;  for i := 0 to \_QForm.sb.ControlCount - 1 do  begin  if \_QForm.sb.Controls[i] is TGMV\_TemplateQualifierBox then  begin  iOrder := \_QForm.sb.ControlCount - 1 - i;  TGMV\_TemplateQualifierBox(\_QForm.sb.Controls[i]).TabOrder :=  iOrder;  end;  end;  for i := 0 to FPanelList.Count - 1 do  with TGMV\_TemplateQualifierBox(FPanelList[i]) do  begin  {  Left := i \* (\_QForm.Width div FPanelList.Count);  Width := \_QForm.Width div FPanelList.Count;  if i < (FPanelList.Count - 1) then  Align := alLeft  else  Align := alClient;  }  Align := alTop;  Visible := True;  OnClick := QualifierClicked;  setPopupLayout;  end;  i := 1;  for j := 0 to FPanelList.Count - 1 do  begin  TQB := TGMV\_TemplateQualifierBox(FPanelList[j]);  s := TQB.DefaultQualifierIEN;  ii := StrToIntDef(s,0);  if ii < 1 then  begin  s := piece(Quals, ':', i);// assigning qualifiers as DEFAULT (!?)  try  if s <> '' then // changed on 050708 by DNS  begin  TGMV\_TemplateQualifierBox(FPanelList[j]).DefaultQualifierIEN := s; // piece(Quals, ':', i);  // TGMV\_TemplateQualifierBox(FPanelList[j]).OnQualClick(TGMV\_TemplateQualifierBox(FPanelList[j]).CLB);  TGMV\_TemplateQualifierBox(FPanelList[j]).OnQualClick(nil);  end  else  // TGMV\_TemplateQualifierBox(FPanelList[j]).OnQualClick(TGMV\_TemplateQualifierBox(FPanelList[j]).CLB);  TGMV\_TemplateQualifierBox(FPanelList[j]).OnQualClick(nil);  except  on E: Exception do  ShowMessage('Error assigning <'+s+'> as the default qualifier'+#13#10+  E.Message);  end;  end;  inc(i);  end;  // --------------------------------Set Default Method for BP if value is like nnn/  // if (VType = vtBP) and (pos('/',aValue) = Length(aValue)) then  // QForm.SetDefaultQualifier('2','45',vtBP);  edtQuals.Text := QualifierNames;  PositionForm(\_QForm);  ShowModal;  if ModalResult = mrOk then  begin  Quals := \_QForm.FQIENS;  QualsDisplay := \_QForm.edtQuals.Text;  Result := True;  end;  finally  free;  end;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| function SelectQualifiers(VType: TVitalType; var Quals, QualsDisplay: string; Ctrl: TControl;aValue:String; aGMV\_Template: TGMV\_TemplateVital): Boolean;  var  \_QForm: TfrmGMV\_Qualifiers;  QPanel: TGMV\_TemplateQualifierBox;  i, j: integer;  pt: TPoint;  RetList**, QualList**: TStrings;  **IsReq: Boolean;**  **LookupIEN: String;**  begin  Result := False;  **//Create the form**  \_QForm := TfrmGMV\_Qualifiers.Create(Application);  with \_QForm do  try  **fGMV\_Template := aGMV\_Template;**  **//set the position relative to the button**  pt := Ctrl.Parent.ClientToScreen(Point(Ctrl.Left, Ctrl.Top));  Left := pt.x;  Top := pt.y + Ctrl.Height;  **//Get the categories**  RetList := getCategoryQualifiers(GMVVitalTypeIEN[VType]);  **try**  **QualList := TStringList.Create;**  **try**  **//Set up the qualifier list**  **QualList.Delimiter := ':';**  **QualList.DelimitedText := Quals;**  **//Set the title**  \_QForm.pnlTitle.Caption := piece(RetList.Strings[0],'^',**2**) + ' Qualifiers';  **//Create the panels for each category (ignore the first return)**  for i := **1** to RetList.Count - 1 do  begin  **LookupIEN := aGMV\_Template.IEN + '\_' + piece(RetList[i], '^', 1);**  **IsReq := aGMV\_Template.IsRequired[LookupIEN];**  QPanel := TGMV\_TemplateQualifierBox.CreateParented(\_QForm,\_QForm.sb, GMVVitalTypeIEN[VType], piece(RetList[i], '^', **1**),''**, IsReq**);  **QPanel.Top := QPanel.Height;**  **QPanel.Visible := True;**  **QPanel.OnClick := QualifierClicked;**  **QPanel.setPopupLayout;**  **//Look for the qualifiers**  **//Check if there is already a default set**  **if StrToIntDef(QPanel.DefaultQualifierIEN, 0) < 1 then**  **begin**  **//Loop through the list of Qualifiers and try to match**  **for J := (QualList.Count - 1) downto 0 do**  **begin**  **try**  **//Check if IEN is in the list**  **If QPanel.IENExist(QualList[J]) then**  **begin**  **QPanel.DefaultQualifierIEN := QualList[J];**  **QualList.Delete(J);**  **break;**  **end;**  **QPanel.OnQualClick(nil);**  **except**  **on E: Exception do**  **ShowMessage('Error assigning <'+QualList[J]+'> as the default qualifier'+#13#10+**  **E.Message);**  **end;**  **end;**  **end;**  **//Add the panel to our list**  FPanelList.Add(QPanel);  end;  **finally**  **QualList.Free;**  **end;**  **Finally**  RetList.Free;  **end;**  edtQuals.Text := QualifierNames;  PositionForm(\_QForm);  ShowModal;  if ModalResult = mrOk then  begin  Quals := \_QForm.FQIENS;  QualsDisplay := \_QForm.edtQuals.Text;  Result := True;  end;  finally  free;  end;  end; |

FormActivate

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | FormActivate | | | |
| **Short Description** | Ensures that the form opens on the proper screen | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_Qualifiers.FormActivate(Sender: TObject);  **Var**  **MaxW, MaxH: Integer;**  begin  **if Screen.MonitorCount = 1 then**  **begin**  **MaxH := Screen.WorkAreaHeight;**  **MaxW := Screen.WorkAreaWidth;**  **end else begin**  **MaxH := Screen.DesktopHeight;**  **MaxW := Screen.DesktopWidth;**  **end;**  if (Left + Width) > **MaxW** then  Left := **MaxW - Width**;  if (Top + Height) > **MaxH** then  Top := **MaxH - Height** - pnlBottom.Height div 2;  end; |

frmGMV\_EnteredInError

Summary of changes

Added new “entered in error” reason.

Detailed Changes

Table 46: Forms

| Forms | Description | | | |
| --- | --- | --- | --- | --- |
| **Form Name** | frmGMV\_EnteredInError | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Form Functionality** |  | | | |

| Current Form Layout |
| --- |
|  |

| Modified Form Layout (Changes are in bold) |
| --- |
|  |

frmGMV\_SupO2

Summary of changes

Allows users to fill out O2 specific qualifiers

Detailed Changes

Uses Clause

Added uGMV\_Template to first uses section.

Table 40: Events

| Name | Type | Description |
| --- | --- | --- |
| Form | **OnCloseQuery** |  |

Table 46: Forms

| Forms | Description | | | |
| --- | --- | --- | --- | --- |
| **Form Name** | frmGMV\_SupO2 | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Form Functionality** |  | | | |

| Current Form Layout |
| --- |
|  |

| Modified Form Layout (Changes are in bold) |
| --- |
|  |

FormCloseQuery

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | FormCloseQuery | | | |
| **Short Description** | Performs check to ensure that required fields are filled out | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_SupO2.FormCloseQuery(Sender: TObject; var CanClose: Boolean);  Const  aMsg = 'Method is required';  begin  CanClose := True;  if fGMV\_Template.IsRequired[fGMV\_Template.IEN +'\_' + fCategory] and (Trim(cbMethod.Text) = '') Then  begin  ShowMessage(aMsg);  CanClose := false;  end;  end; |

GetSupplementO2Data

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | GetSupplementO2Data | | | |
| **Short Description** | Change sets the required label as needed | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: FlowRate | | | | |
| Definition: string  Name: Percentage  Definition: string  Name: ctrl  Definition: TControl  Name: sQual  Definition: string | | | | |
| **Output Attribute Name and Definition** | Name: Result | | | | |
| Definition: Boolean | | | | |

| Current Logic |
| --- |
| function GetSupplementO2Data(var FlowRate: string; var Percentage: string; ctrl: TControl;sQual:String): Boolean;  var  s:String;  sType, sCategory: String;  i: INteger;  pt: TPoint;  SL: TStringList;  begin  Result := False;  with TfrmGMV\_SupO2.Create(Application) do  try  pt := Ctrl.Parent.ClientToScreen(Point(Ctrl.Left, Ctrl.Top));  Left := pt.x;  Top := pt.y + Ctrl.Height;  edtFlow.Text := FlowRate;  edtO2Con.Text := Percentage;  sType := getVitalTypeIEN('PULSE OXIMETRY');  sCategory := getVitalCategoryIEN('METHOD');  SL := getQualifiers(sType,sCategory);  if SL.Count > 1 then  begin  if pos('-None-',cbMethod.Text) = 0 then  cbMethod.Items.Add('-None-'); //CQ Ticket #6942  for i := 1 to SL.Count - 1 do  begin  s := SL[i];  cbMethod.Items.AddObject(TitleCase(Piece(s,'^', 2)),  Pointer(StrToIntDef(Piece(s,'^', 1), 0)));  end;  end;  SL.Free;  s := Piece(sQual,'[',2);  if pos(' ',s)= 0 then  s := Piece(s,']',1)  else  s := Piece(s,' ',1);  i := cbMethod.Items.IndexOf(s);  if i >= 0 then  begin  cbMethod.ItemIndex := i;  QualVal := integer(cbMethod.Items.Objects[i]); //AAN 11/5/2002 initial value added;  end;  ActiveControl := edtFlow;  ShowModal;  if ModalResult = mrOk then  begin  FlowRate := edtFlow.Text;  //AAN 07/09/2002 Percentage := edtO2Con.text;  Percentage := edtO2Con.text + '^'+IntToStr(QualVal)+'^'+cbMethod.Text;  Result := True;  end;  finally  free;  end;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| function GetSupplementO2Data(var FlowRate: string; var Percentage: string; ctrl: TControl;sQual:String**; aGMV\_Template: TGMV\_TemplateVital**): Boolean;  var  s:String;  sType: String;  i: INteger;  pt: TPoint;  SL: TStringList;  begin  Result := False;  with TfrmGMV\_SupO2.Create(Application) do  try  pt := Ctrl.Parent.ClientToScreen(Point(Ctrl.Left, Ctrl.Top));  Left := pt.x;  Top := pt.y + Ctrl.Height;  edtFlow.Text := FlowRate;  edtO2Con.Text := Percentage;  sType := getVitalTypeIEN('PULSE OXIMETRY');  **fCategory := getVitalCategoryIEN('METHOD');**  **fGMV\_Template := aGMV\_Template;**  **if aGMV\_Template.IsRequired[aGMV\_Template.IEN +'\_' + fCategory] then**  **lblMethodValue.Caption := '\*Method';**  SL := getQualifiers(sType,fCategory);  if SL.Count > 1 then  begin  if pos('-None-',cbMethod.Text) = 0 then  cbMethod.Items.Add('-None-'); //CQ Ticket #6942  for i := 1 to SL.Count - 1 do  begin  s := SL[i];  cbMethod.Items.AddObject(TitleCase(Piece(s,'^', 2)),  Pointer(StrToIntDef(Piece(s,'^', 1), 0)));  end;  end;  SL.Free;  s := Piece(sQual,'[',2);  if pos(' ',s)= 0 then  s := Piece(s,']',1)  else  s := Piece(s,' ',1);  i := cbMethod.Items.IndexOf(s);  if i >= 0 then  begin  cbMethod.ItemIndex := i;  QualVal := integer(cbMethod.Items.Objects[i]); //AAN 11/5/2002 initial value added;  end;  ActiveControl := edtFlow;  ShowModal;  if ModalResult = mrOk then  begin  FlowRate := edtFlow.Text;  //AAN 07/09/2002 Percentage := edtO2Con.text;  Percentage := edtO2Con.text + '^'+IntToStr(QualVal)+'^'+cbMethod.Text;  Result := True;  end;  finally  free;  end;  end; |

cbMethodChange

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | cbMethodChange | | | |
| **Short Description** | Change cleans up handled access violation that did not have to happen | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: FlowRate | | | | |
| Name: Sender  Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| procedure TfrmGMV\_SupO2.cbMethodChange(Sender: TObject);  function getQualVal: Integer;  begin  try  Result := integer(cbMethod.Items.Objects[cbMethod.ItemIndex]);  except  Result := -99;  end;  end;  procedure setFields(anEnabled:Boolean);  var  \_Color: TColor;  begin  if anEnabled then \_Color := clWindowText  else  begin  \_Color := clGrayText;  // there should be no values of concentration for Room Air  edtFlow.Text := '';  edtO2Con.Text := '';  end;  edtFlow.Enabled := anEnabled;  edtO2Con.Enabled := anEnabled;  udFlow.Enabled := anEnabled;  udO2.Enabled := anEnabled;  lblFlow.Font.Color := \_Color;  lblO2Con.Font.Color := \_Color;  lblLitMin.Font.Color := \_Color;  lblPercent.Font.Color := \_Color;  QualVal := getQualVal;  end;  begin  if (cbMethod.ItemIndex = 0) // CQ Ticket # 6942 - Start  // or (pos('ROOM AIR',uppercase(cbMethod.Text))>0)  then  setFields(False)  else // CQ Ticket # 6942 - end  setFields(True);  end; |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_SupO2.cbMethodChange(Sender: TObject);  function getQualVal: Integer;  begin  **if cbMethod.ItemIndex > 0 then**  **begin**  try  Result := integer(cbMethod.Items.Objects[cbMethod.ItemIndex]);  except  Result := -99;  end;  **end else**  **Result := -99;**  end;  procedure setFields(anEnabled:Boolean);  var  \_Color: TColor;  begin  if anEnabled then \_Color := clWindowText  else  begin  \_Color := clGrayText;  // there should be no values of concentration for Room Air  edtFlow.Text := '';  edtO2Con.Text := '';  end;  edtFlow.Enabled := anEnabled;  edtO2Con.Enabled := anEnabled;  udFlow.Enabled := anEnabled;  udO2.Enabled := anEnabled;  lblFlow.Font.Color := \_Color;  lblO2Con.Font.Color := \_Color;  lblLitMin.Font.Color := \_Color;  lblPercent.Font.Color := \_Color;  QualVal := getQualVal;  end;  begin  if (cbMethod.ItemIndex = 0) // CQ Ticket # 6942 - Start  // or (pos('ROOM AIR',uppercase(cbMethod.Text))>0)  then  setFields(False)  else // CQ Ticket # 6942 - end  setFields(True);  end; |

fGMV\_InputLite

Summary of changes

Main form used to fill out vital measurements.

Detailed Changes

acSaveInputExecute

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | acSaveInputExecute | | | |
| **Short Description** | Change checks to verify required fields (if applicable) are filled out. Allso fixes handled access violation | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| procedure TfrmGMV\_InputLite.acSaveInputExecute(Sender: TObject);  const  sLastProcessedMessage = 'The last patient in the list processed' + #13 + 'Close the input window?';  sSelectHospital = 'Please select a valid hospital location for this patient.';  var  aTime: TDateTime;  sNoDataEntered: String;  DT: TDateTime;  procedure AssignImageIndex(anIndex: Integer);  begin  if lvSelPatients.ItemFocused.SubItems[3] = '' then  begin  lvSelPatients.ItemFocused.ImageIndex := anIndex;  lvSelPatients.ItemFocused.SubItems[3] := IntToStr(anIndex);  end  else  begin  lvSelPatients.ItemFocused.SubItems[3] := lvSelPatients.ItemFocused.SubItems[3] + ' ' + IntToStr(anIndex);  lvSelPatients.ItemFocused.ImageIndex := indMultipleInput;  end;  end;  function CheckData(aDT: TDateTime): Boolean;  var  i: Integer;  begin  for i := 0 to sbxMain.ComponentCount - 1 do  if not TfraGMV\_InputOne2(sbxMain.Components[i]).Check(aDT) then  break;  Result := i = sbxMain.ComponentCount;  if not Result then  begin  TfraGMV\_InputOne2(sbxMain.Components[i]).cbxInput.SelStart := 0;  TfraGMV\_InputOne2(sbxMain.Components[i]).cbxInput.SelLength := Length(TfraGMV\_InputOne2(sbxMain.Components[i]).cbxInput.Text);  TfraGMV\_InputOne2(sbxMain.Components[i]).cbxInput.SetFocus;  end;  end;  procedure ProcessNext(ResultIndex: Integer);  begin  if (lvSelPatients = nil) or (lvSelPatients.Items.Count < 2) then  begin  if bSlave then // DLL  begin  if chkbCloseOnSave.Checked or (Sender = btnSaveAndExit) // added DNS 050707 for DLL use  then  begin  Close;  Exit; // Get out of here now, we just triggered the form Destroy with caFree  end  else  UpdateLatestVitals(FDFN, True);  end  else  begin  // Do we need to close window if the list contains only one patient?  // Yes we do  // Do we need to have a message?  // No we don't  // s := LastProcessedMessage;  // if MessageDlgS(S,mtInformation,mbOKCancel,0) = mrOK then  if Sender = btnSaveAndExit then  ModalResult := mrOk;  end;  end;  try  if lvSelPatients.ItemFocused.Index < 0 then  lvSelPatients.ItemFocused := lvSelPatients.Items[0];  except  lvSelPatients.ItemFocused := lvSelPatients.Items[0];  end;  try  if lvSelPatients.ItemFocused.Index < 0 then  lvSelPatients.ItemFocused := lvSelPatients.Items[0];  if lvSelPatients.ItemFocused.Index < lvSelPatients.Items.Count - 1 then  begin  AssignImageIndex(ResultIndex);  lvSelPatients.ItemFocused := lvSelPatients.Items[lvSelPatients.ItemFocused.Index + 1];  lvSelPatients.ItemFocused.ImageIndex := 1;  lvSelPatientsClick(Sender);  CleanUpVitalsList;  SetVitalsList(FNewTemplate + '^' + FNewTemplateDescription); // DNS  if not lvSelPatients.CheckBoxes then  ckbOnPass.Checked := False;  FocusInputField;  end  else  begin  AssignImageIndex(ResultIndex);  lvSelPatients.Invalidate;  // lvSelPatientsClick(Sender);// may be it is too much  if lvSelPatients.Items.Count < 2 then  ModalResult := mrOk  else if MessageDLG(sLastProcessedMessage, mtInformation, mbOKCancel, 0) = mrOk then  ModalResult := mrOk;  end;  except  end;  end;  begin  // Check all input panels for valid values  DT := WindowsDateTimeToFMDateTime(trunc(dtpDate.Date) + (dtpTime.DateTime - trunc(dtpTime.Date)));  if not CheckData(DT) then  exit;  // Hospital location should be set before saving data -- 03/14/2003 AAN  if (\_FHospitalIEN = '') or (\_FHospitalIEN = '0') then  MessageDLG(sSelectHospital, mtWarning, [mbOK], 0)  else  begin  if InputString <> '' then  begin  aTime := EventStart('Save Vitals -- Begin');  SaveData(DT);  EventStop('Save Vitals -- End', '', aTime);  if ckbOnPass.Checked then  ProcessNext(indOnPass)  else  ProcessNext(indSingleInput);  end  else  begin  sNoDataEntered := 'No data entered for patient ' + #13 + getPatientName + #13;  case lvSelPatients.Items.Count of  0, 1:  begin  if (MessageDLG(sNoDataEntered + 'Close the window?', mtInformation, [mbYes, mbNo], 0) = mrYes) then  if bSlave then  Close  else  ModalResult := mrOk;  end;  else  begin  if (MessageDLG(sNoDataEntered + 'Process next Patient?', mtInformation, [mbYes, mbNo], 0) = mrYes) then  ProcessNext(indBlankInput);  end;  end;  end;  end;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfrmGMV\_InputLite.acSaveInputExecute(Sender: TObject);  const  sLastProcessedMessage = 'The last patient in the list processed' + #13 + 'Close the input window?';  sSelectHospital = 'Please select a valid hospital location for this patient.';  var  aTime: TDateTime;  sNoDataEntered: String;  DT: TDateTime;  procedure AssignImageIndex(anIndex: Integer);  begin  if lvSelPatients.ItemFocused.SubItems[3] = '' then  begin  lvSelPatients.ItemFocused.ImageIndex := anIndex;  lvSelPatients.ItemFocused.SubItems[3] := IntToStr(anIndex);  end  else  begin  lvSelPatients.ItemFocused.SubItems[3] := lvSelPatients.ItemFocused.SubItems[3] + ' ' + IntToStr(anIndex);  lvSelPatients.ItemFocused.ImageIndex := indMultipleInput;  end;  end;  function CheckData(aDT: TDateTime): Boolean;  **const**  **ErrDisplayMsg = 'The following error(s) occurred:' + #10#13 + '%s' + #10#13 + 'Please correct and try again';**  var  i**, FirstItem: Integer;**  **aMsg: String;**  **aErrMsg: TStringList;**  begin  for i := 0 to sbxMain.ComponentCount - 1 do  if not TfraGMV\_InputOne2(sbxMain.Components[i]).Check(aDT) then  break;  Result := i = sbxMain.ComponentCount;  if not Result then  begin  TfraGMV\_InputOne2(sbxMain.Components[i]).cbxInput.SelStart := 0;  TfraGMV\_InputOne2(sbxMain.Components[i]).cbxInput.SelLength :=  Length(TfraGMV\_InputOne2(sbxMain.Components[i]).cbxInput.Text);  TfraGMV\_InputOne2(sbxMain.Components[i]).cbxInput.SetFocus;  **end else begin**  **//Check the qualifiers**  **aErrMsg := TStringList.Create;**  **try**  **FirstItem := -1;**  **for i := 0 to sbxMain.ComponentCount - 1 do**  **begin**  **if not TfraGMV\_InputOne2(sbxMain.Components[i]).CheckQualifer(aMsg) then**  **begin**  **//Create the error message string list**  **if FirstItem = -1 then**  **FirstItem := I;**  **aErrMsg.Add(aMsg);**  **end;**  **end;**  **//Errors present, show message and halt**  **if FirstItem <> -1 then**  **begin**  **Result := False;**  **MessageDlg(Format(ErrDisplayMsg, [aErrMsg.Text]), mtError, [mbOk], -1 );**  **TfraGMV\_InputOne2(sbxMain.Components[FirstItem]).bbtnQualifiers.Click;**  **end;**  **finally**  **FreeAndNil(aErrMsg);**  **end;**  end;  end;  procedure ProcessNext(ResultIndex: Integer);  begin  if (lvSelPatients = nil) or (lvSelPatients.Items.Count < 2) then  begin  if bSlave then // DLL  begin  if chkbCloseOnSave.Checked or (Sender = btnSaveAndExit) // added DNS 050707 for DLL use  then  begin  Close;  Exit; // Get out of here now, we just triggered the form Destroy with caFree  end  else  UpdateLatestVitals(FDFN, True);  end  else  begin  // Do we need to close window if the list contains only one patient?  // Yes we do  // Do we need to have a message?  // No we don't  // s := LastProcessedMessage;  // if MessageDlgS(S,mtInformation,mbOKCancel,0) = mrOK then  if Sender = btnSaveAndExit then  ModalResult := mrOk;  end;  end;  **{**try  if lvSelPatients.ItemFocused.Index < 0 then  lvSelPatients.ItemFocused := lvSelPatients.Items[0];  except  lvSelPatients.ItemFocused := lvSelPatients.Items[0];  end; **}**  if Assigned(lvSelPatients.ItemFocused) then  begin  try  if lvSelPatients.ItemFocused.Index < 0 then  lvSelPatients.ItemFocused := lvSelPatients.Items[0];  if lvSelPatients.ItemFocused.Index < lvSelPatients.Items.Count - 1 then  begin  AssignImageIndex(ResultIndex);  lvSelPatients.ItemFocused := lvSelPatients.Items[lvSelPatients.ItemFocused.Index + 1];  lvSelPatients.ItemFocused.ImageIndex := 1;  lvSelPatientsClick(Sender);  CleanUpVitalsList;  SetVitalsList(FNewTemplate + '^' + FNewTemplateDescription); // DNS  if not lvSelPatients.CheckBoxes then  ckbOnPass.Checked := False;  FocusInputField;  end  else  begin  AssignImageIndex(ResultIndex);  lvSelPatients.Invalidate;  // lvSelPatientsClick(Sender);// may be it is too much  if lvSelPatients.Items.Count < 2 then  ModalResult := mrOk  else if MessageDLG(sLastProcessedMessage, mtInformation, mbOKCancel, 0) = mrOk then  ModalResult := mrOk;  end;  except  end;  end;  **end;**  begin  // Check all input panels for valid values  DT := WindowsDateTimeToFMDateTime(trunc(dtpDate.Date) + (dtpTime.DateTime - trunc(dtpTime.Date)));  if not CheckData(DT) then  exit;  // Hospital location should be set before saving data -- 03/14/2003 AAN  if (\_FHospitalIEN = '') or (\_FHospitalIEN = '0') then  MessageDLG(sSelectHospital, mtWarning, [mbOK], 0)  else  begin  if InputString <> '' then  begin  aTime := EventStart('Save Vitals -- Begin');  SaveData(DT);  EventStop('Save Vitals -- End', '', aTime);  if ckbOnPass.Checked then  ProcessNext(indOnPass)  else  ProcessNext(indSingleInput);  end  else  begin  sNoDataEntered := 'No data entered for patient ' + #13 + getPatientName + #13;  case lvSelPatients.Items.Count of  0, 1:  begin  if (MessageDLG(sNoDataEntered + 'Close the window?', mtInformation, [mbYes, mbNo], 0) = mrYes) then  if bSlave then  Close  else  ModalResult := mrOk;  end;  else  begin  if (MessageDLG(sNoDataEntered + 'Process next Patient?', mtInformation, [mbYes, mbNo], 0) = mrYes) then  ProcessNext(indBlankInput);  end;  end;  end;  end;  end; |

mGMV\_InputOne2

Summary of changes

Holds the data for each vital measurement to be used on InputLite.

Detailed Changes

Table 44: Class Properties

| Class Properties Name | Type | Visibility | Description |
| --- | --- | --- | --- |
| CheckErrorMessage | String | Public | Returns the error message produced from the check function |

SetRequireIndicator

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | SetRequireIndicator | | | |
| **Short Description** | Sets the label for the methods to required (if applicable) | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| Procedure TfraGMV\_InputOne2.SetRequireIndicator;  begin  //Only add the \* if there is a value entered  if (FTemplateVital.IsRequired[FTemplateVital.IEN]) and (cbxInput.Text <> '') and (Pos('\*', lblQualifiers.Caption) < 1) then  lblQualifiers.Caption := '\*' + lblQualifiers.Caption  else begin  //strip the \* if no text  if (Pos('\*', lblQualifiers.Caption) > 0) and (cbxInput.Text = '') then  lblQualifiers.Caption := Copy(lblQualifiers.Caption, Pos('\*', lblQualifiers.Caption) + 1, length(lblQualifiers.Caption));  end;  end; |

CheckQualifer

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | CheckQualifer | | | |
| **Short Description** | Sets the label for the methods to required (if applicable) | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aRtnMsg | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: aRtnMsg | | | | |
| Definition: String  Name: Result  Definition: Boolean | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| Function TfraGMV\_InputOne2.CheckQualifer(var aRtnMsg: String): Boolean;  Const  aQualMissing = '%s requires that qualifers be entered.';  begin  aRtnMsg := '';  Result := True;  if cbxInput.Text <> '' then  begin  //Check for required qualifiers  if (fTemplateVital.IsRequired[fTemplateVital.IEN]) and (Trim(fVitalQualifiers) = '') then  begin  aRtnMsg := Format(aQualMissing, [TitleCase(FTemplateVital.VitalName)]);  Result := False;  end;  end;  end; |

bbtnQualifiersClick

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | bbtnQualifiersClick | | | |
| **Short Description** | Opens the qualifier form for a given Vital Measuremement | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObejct | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| procedure TfraGMV\_InputOne2.bbtnQualifiersClick(Sender: TObject);  var  s: String;  Quals: string;  QualsText: string;  PO2FlowRate: string;  PO2Percentage: string;  //const // DNS 050714  // cRoomAir = 'ROOM AIR'; // DNS 050714  begin  if FTemplateVitalType = vtPO2 then  begin  PO2FlowRate := FPO2FlowRate;  PO2Percentage := FPO2Percentage;  if GetSupplementO2Data(PO2FlowRate, PO2Percentage, TControl(Sender),lblQualifiers.Caption) then  begin  FPO2FlowRate := PO2FlowRate;  FPO2Percentage := piece(PO2Percentage,'^',1);  FVitalQualifiers := piece(PO2Percentage,'^',2);  QualsText := piece(PO2Percentage,'^',3);  s := ' ';//AAN 11/07/2002 -- two spaces between Qualifier and data  if FPO2FlowRate <> '' then  s := s + ' ' + FPO2FlowRate + ' l/min';  if FPO2Percentage <> '' then  s := s + ' ' + FPO2Percentage + ' %';  if s = ' ' then s := '';  // 20071220 DNS --- all flow rates and %% values are included --- /////  lblQualifiers.Caption := '[' + QualsText + s + ']'  (\* ==================================================================== 20071220  if (FPO2FlowRate <> '') or (FPO2Percentage <> '') then  lblQualifiers.Caption := '[' + QualsText + s + ']'  else if pos(cRoomAir,uppercase(QualsText)) > 0 then // DNS 050714  lblQualifiers.Caption := '[' + QualsText + ']' // DNS 050714  else  lblQualifiers.Caption := '[]';  ==============================================================================\*)  end;  Exit;  end;  Quals := FVitalQualifiers;  if SelectQualifiers(FTemplateVitalType, Quals, QualsText, TControl(Sender),cbxInput.Text) then  begin  FVitalQualifiers := Quals;  lblQualifiers.Caption := '[' + QualsText + ']';  end;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfraGMV\_InputOne2.bbtnQualifiersClick(Sender: TObject);  var  s: String;  Quals: string;  QualsText: string;  PO2FlowRate: string;  PO2Percentage: string;  //const // DNS 050714  // cRoomAir = 'ROOM AIR'; // DNS 050714  begin  if FTemplateVitalType = vtPO2 then  begin  PO2FlowRate := FPO2FlowRate;  PO2Percentage := FPO2Percentage;  if GetSupplementO2Data(PO2FlowRate, PO2Percentage, TControl(Sender),lblQualifiers.Caption**, fTemplateVital**) then  begin  FPO2FlowRate := PO2FlowRate;  FPO2Percentage := piece(PO2Percentage,'^',1);  FVitalQualifiers := piece(PO2Percentage,'^',2);  QualsText := piece(PO2Percentage,'^',3);  s := ' ';//AAN 11/07/2002 -- two spaces between Qualifier and data  if FPO2FlowRate <> '' then  s := s + ' ' + FPO2FlowRate + ' l/min';  if FPO2Percentage <> '' then  s := s + ' ' + FPO2Percentage + ' %';  if s = ' ' then s := '';  // 20071220 DNS --- all flow rates and %% values are included --- /////  lblQualifiers.Caption := '[' + QualsText + s + ']'  (\* ==================================================================== 20071220  if (FPO2FlowRate <> '') or (FPO2Percentage <> '') then  lblQualifiers.Caption := '[' + QualsText + s + ']'  else if pos(cRoomAir,uppercase(QualsText)) > 0 then // DNS 050714  lblQualifiers.Caption := '[' + QualsText + ']' // DNS 050714  else  lblQualifiers.Caption := '[]';  ==============================================================================\*)  end;  Exit;  end;  Quals := FVitalQualifiers;  if SelectQualifiers(FTemplateVitalType, Quals, QualsText, TControl(Sender),cbxInput.Text**, fTemplateVital**) then  begin  FVitalQualifiers := Quals;  lblQualifiers.Caption := '[' + QualsText + ']';  **SetRequireIndicator;**  end;  end; |

CheckQualifer

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | cbxInputChange | | | |
| **Short Description** | Used when text is enterd into the value field | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: Sender | | | | |
| Definition: TObject | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| procedure TfraGMV\_InputOne2.cbxInputChange(Sender: TObject);  begin  if (cbxInput.Text = cRefused) or (cbxInput.Text = cUnavailable) then  SetPanelStatus(False)  else  SetPanelStatus(True);  GetParentForm(self).Perform(CM\_VITALCHANGED,0,0);//09/11/02  end; |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TfraGMV\_InputOne2.cbxInputChange(Sender: TObject);  begin  if (cbxInput.Text = cRefused) or (cbxInput.Text = cUnavailable) then  SetPanelStatus(False)  else  SetPanelStatus(True);  GetParentForm(self).Perform(CM\_VITALCHANGED,0,0);//09/11/02  **SetRequireIndicator;**  end; |

uGMV\_QualifyBox

Summary of changes

This holds the logic for the qualifier boxes

IENExist

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | IENExist | | | |
| **Short Description** | Checks to see if the qualifier already exist | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aIEN | | | | |
| Definition: Integer | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| function TGMV\_TemplateQualifierBox.IENExist(aIEN: String): Boolean;  Var  s:String;  i: Integer;  begin  Result := false;  for i := 1 to fQualifierItems.Count - 1 do  begin  s := fQualifierItems[i];  if piece(s,'^',1) = aIEN then  begin  Result := True;  break;  end;  end;  end; |

getIENByName

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | getIENByName | | | |
| **Short Description** | Returns the IEN based on a qualifier name | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aName | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: Result | | | | |
| Definition: String | | | | |

| Current Logic |
| --- |
| function TGMV\_TemplateQualifierBox.getIENByName(anIEN:String):String;  var  s:String;  i: Integer;  begin  Result := '';  for i := 1 to fQualifierItems.Count - 1 do  begin  s := fQualifierItems[i];  if piece(s,'^',2) = uppercase(fDefaultQualifierName) then  begin  Result := piece(s,'^',1);  break;  end;  end;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| function TGMV\_TemplateQualifierBox.getIENByName(a**Name**:String):String;  var  s:String;  i: Integer;  begin  Result := '';  for i := 1 to fQualifierItems.Count - 1 do  begin  s := fQualifierItems[i];  if piece(s,'^',2) = uppercase(**aName**) then  begin  Result := piece(s,'^',1);  break;  end;  end;  end; |

CreateParented

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | CreateParented | | | |
| **Short Description** | Creates the qualifier entry screen | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: anOwner | | | | |
| Definition: TWinControl  Name: aParent  Definition: TWinControl  Name: VitalIEN  Definition: string  Name: CategoryIEN  Definition: string  Name: DefaultIEN  Definition: string  Name: Required  Definition: Boolean  Name: ViewMode  Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| constructor TGMV\_TemplateQualifierBox.CreateParented(anOwner,aParent: TWinControl;  VitalIEN, CategoryIEN, DefaultIEN: string;ViewMode:String='');  var  SL: TStringList;  procedure setRadioGroup;  var  iDefault,  i: integer;  SB: TScrollBox;  begin  SB := TScrollBox.Create(self);  SB.Parent := self;  SB.Align := alClient;  fRG := TRadioGroup.Create(self);  with fRG do  begin  Caption := TitleCase(GMVCats.Entries[GMVCats.IndexOfIEN(CategoryIEN)]);  Parent := SB;  Color := clWindow;  Align := alClient;  iDefault := 0;  for i := 0 to fQualifierItems.Count - 1 do  begin  Items.Add(TitleCase(Piece(fQualifierItems[i], '^', 2)));  if Piece(fQualifierItems[i], '^', 1)=DefaultIEN then  iDefault := i;  end;  ItemIndex := iDefault;  OnClick := OnQualClick;  OnEnter := OnQualEnter;  OnExit := OnQualExit;  end;  end;  procedure setDropDownList;  var  iDefault,  i: integer;  begin  Height := 22;  ParentFont := False;  lblName := TLabel.Create(AnOwner);  lblName.Caption := TitleCase(GMVCats.Entries[GMVCats.IndexOfIEN(CategoryIEN)]);  lblName.Top := 4;  lblName.Left := 8;  lblName.Parent := self;  fcb := TComboBox.Create(Self);  fcb.Left := 80;  fcb.Top := 2;  fcb.Parent := Self;  iDefault := -1;  for i := 0 to fQualifierItems.Count - 1 do  begin  fcb.Items.Add(TitleCase(Piece(fQualifierItems[i], '^', 2)));  if Piece(fQualifierItems[i], '^', 1)=DefaultIEN then  iDefault := i;  end;  fcb.ItemIndex := iDefault;  fcb.OnChange := OnQualClick;  fCB.OnEnter := OnQualEnter;  fCB.OnExit:= OnQualExit;  lblName.FocusControl := fCB;  end;  begin  inherited Create(anOwner);  fQualifierItems := TStringList.Create;  Visible := False;  Parent := aParent;  Align := alTop;  BevelInner := bvNone;  BevelOuter := bvNone;  Caption := ' '+TitleCase(GMVCats.Entries[GMVCats.IndexOfIEN(CategoryIEN)]);  FVitalIEN := VitalIEN;  FCategoryIEN := CategoryIEN;  FDefaultQualifierName := DefaultIEN;  SL := getQualifiers(VitalIEN,CategoryIEN);  fQualifierItems.Text := SL.Text;  if fQualifierItems.Count > 0 then  fQualifierItems[0] := '-None-';  SL.Free;  fViewMode := ViewMode;  if fViewMode = '' then  setDropDownList  else  setRadioGroup;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| constructor TGMV\_TemplateQualifierBox.CreateParented(anOwner,aParent: TWinControl;  VitalIEN, CategoryIEN, DefaultIEN: string**; Required: Boolean**; ViewMode:String='');  var  SL: TStringList;  procedure setRadioGroup;  var  iDefault,  i: integer;  SB: TScrollBox;  begin  SB := TScrollBox.Create(self);  SB.Parent := self;  SB.Align := alClient;  fRG := TRadioGroup.Create(self);  with fRG do  begin  Caption := TitleCase(GMVCats.Entries[GMVCats.IndexOfIEN(CategoryIEN)]);  **if Required then**  **Caption := '\*' + Trim(Caption);**  Parent := SB;  Color := clWindow;  Align := alClient;  iDefault := 0;  for i := 0 to fQualifierItems.Count - 1 do  begin  Items.Add(TitleCase(Piece(fQualifierItems[i], '^', 2)));  if Piece(fQualifierItems[i], '^', 1)=DefaultIEN then  iDefault := i;  end;  ItemIndex := iDefault;  OnClick := OnQualClick;  OnEnter := OnQualEnter;  OnExit := OnQualExit;  end;  end;  procedure setDropDownList;  var  iDefault,  i: integer;  begin  Height := 22;  ParentFont := False;  lblName := TLabel.Create(AnOwner);    lblName.Caption := TitleCase(GMVCats.Entries[GMVCats.IndexOfIEN(CategoryIEN)]);  **if Required then**  **lblName.Caption := '\*' + Trim(Caption);**  **lblName.align := alLeft;**  **lblName.AlignWithMargins := true;**  **lblName.Margins.Top := 4;**  **lblName.Margins.Left := 8;**  **lblName.Margins.Right := 0;**  **lblName.Margins.Bottom := 0;**  **lblName.Width := 80;**  **lblName.AutoSize := false;**  **//**lblName.Top := 4;  **//**lblName.Left := 8;  lblName.Parent := self;  fcb := TComboBox.Create(Self);  **fcb.align := alClient;**  **fcb.AlignWithMargins := true;**  **fcb.Margins.Top := 2;**  **fcb.Margins.Left := 0;**  **fcb.Margins.Right := 10;**  **fcb.Margins.Bottom := 0;**  **//**fcb.Left := 80;  **//** fcb.Top := 2;  fcb.Parent := Self;  iDefault := -1;  for i := 0 to fQualifierItems.Count - 1 do  begin  fcb.Items.Add(TitleCase(Piece(fQualifierItems[i], '^', 2)));  if Piece(fQualifierItems[i], '^', 1)=DefaultIEN then  iDefault := i;  end;  fcb.ItemIndex := iDefault;  fcb.OnChange := OnQualClick;  fCB.OnEnter := OnQualEnter;  fCB.OnExit:= OnQualExit;  lblName.FocusControl := fCB;  end;  begin  inherited Create(anOwner);  fQualifierItems := TStringList.Create;  Visible := False;  Parent := aParent;  Align := alTop;  BevelInner := bvNone;  BevelOuter := bvNone;  Caption := ' '+TitleCase(GMVCats.Entries[GMVCats.IndexOfIEN(CategoryIEN)]);  FVitalIEN := VitalIEN;  FCategoryIEN := CategoryIEN;  FDefaultQualifierName := DefaultIEN;  SL := getQualifiers(VitalIEN,CategoryIEN);  fQualifierItems.Text := SL.Text;  if fQualifierItems.Count > 0 then  fQualifierItems[0] := '-None-';  SL.Free;  fViewMode := ViewMode;  if fViewMode = '' then  setDropDownList  else  setRadioGroup;  end; |

uGMV\_Template

Summary of changes

This holds the logic for templates

Table 38: GUI Classes

| GUI Classes | Instructions |
| --- | --- |
| **Class Name** | TGMV\_QualReqRec |
| **Derived From Class** | Record |
| **Purpose** | Holds vital measurement and qualifiers ID and requirement setting |

Table 44: Class Properties

| Class Properties Name | Type | Visibility | Description |
| --- | --- | --- | --- |
| IsRequired | Boolean | Public | Return Boolean if IEN passed in is required |
| IsSystemWideRequired | Boolean | Public | Return Boolean if current Vital Measurement IEN is system wide required |
| RequiredAsString | String | Public | Retrun the Vitals/Qualifiers and their requirement settings for the database |
| Required | TGMV\_QualRequireType | Public | Return QualRequireType if IEN passed in is required |

SetRequiredString

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | SetRequiredString | | | |
| **Short Description** | Setter for RequiredString property | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aValue | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TGMV\_TemplateVital.SetRequiredString(const aValue: String);  Var  i: integer;  aTmpLst: TStringList;  aIEN: String;  aReqVal: TGMV\_QualRequireType;  begin  if Trim(aValue) = '' then  begin  // If nothing passed insystem wide then set  if IsQualifierGlobalRequired(FIEN) then  Required[FIEN] := qrSystemWide;  end  else  begin  aTmpLst := TStringList.Create;  try  aTmpLst.CommaText := aValue;  for i := 0 to aTmpLst.Count - 1 do  begin  aIEN := Piece(aTmpLst.Strings[i], '~', 1);  aReqVal := TGMV\_QualRequireType  (StrToIntDef(Piece(aTmpLst.Strings[i], '~', 2), 0));  Required[aIEN] := aReqVal;  { if IsQualifierGlobalRequired(aIEN) then  begin  if aReqVal <> qrTemplateBypass then  Required[FIEN] := qrSystemWide;  end else begin  if aReqVal = qrTemplateBypass then  Required[FIEN] := qrNo;  end; }  end;  finally  aTmpLst.Free;  end;  end;  end; |

GetIsRequired

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | GetIsRequired | | | |
| **Short Description** | Getter for IsRequired property | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aIEN | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| function TGMV\_TemplateVital.GetIsRequired(aIEN: String): Boolean;  begin  result := IsRequiredType(Required[aIEN]);  end; |

IsRequiredType

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | IsRequiredType | | | |
| **Short Description** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aType | | | | |
| Definition: TGMV\_QualRequireType | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| Function TGMV\_TemplateVital.IsRequiredType(aType: TGMV\_QualRequireType): Boolean;  begin  case aType of  qrSystemWide: Result := true;  qrTemplateOnly: Result := true;  qrTemplateBypass: Result := False;  qrNo: Result := False;  else Result := false; //fail safe  end;  end; |

GetIsSystemWideRequired

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | GetIsSystemWideRequired | | | |
| **Short Description** | Getter for IsSystemWideRequired property | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| function TGMV\_TemplateVital.GetIsSystemWideRequired: Boolean;  begin  case Required[FIEN] of  qrSystemWide: Result := true;  else Result := false; //fail safe  end;  end; |

GetRequiredString

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | GetRequiredString | | | |
| **Short Description** | Setter for RequiredString property | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| function TGMV\_TemplateVital.GetRequiredString: String;  Var  I: integer;  begin  Result := '';  for I := Low(FRequiredArray) to High(FRequiredArray) do  begin  if I <> Low(FRequiredArray) then  Result := Result + ',';  Result := Result + FRequiredArray[i].IEN + '~' + IntToStr(Ord(FRequiredArray[i].Required));  end;  end; |

GetRequired

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | GetRequired | | | |
| **Short Description** | Getter for Required property | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aIEN | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| function TGMV\_TemplateVital.GetRequired(aIEN: String): TGMV\_QualRequireType;  Var  I: integer;  IsGlobal, FndRec: Boolean;  begin  Result := qrNo;  FndRec := false;  IsGlobal := IsQualifierGlobalRequired(aIEN);  for I := Low(FRequiredArray) to High(FRequiredArray) do  begin  if FRequiredArray[i].IEN = aIEN then  begin  if IsGlobal then  begin  if FRequiredArray[i].Required <> qrTemplateBypass then  FRequiredArray[i].Required := qrSystemWide;  end else begin  if FRequiredArray[i].Required = qrTemplateBypass then  FRequiredArray[i].Required := qrNo;  end;  Result := FRequiredArray[i].Required;  FndRec := True;  break;  end;  end;  if not FndRec and IsGlobal then  begin  Required[aIen] := qrSystemWide;  Result := qrSystemWide;  end;  end; |

SetRequired

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | SetRequired | | | |
| **Short Description** | Getter for Required property | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aIEN | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| procedure TGMV\_TemplateVital.SetRequired(aIEN: String;  const Value: TGMV\_QualRequireType);  Var  I: integer;  RecRnd: Boolean;  begin  RecRnd := False;  for I := Low(FRequiredArray) to High(FRequiredArray) do  begin  if FRequiredArray[i].IEN = aIEN then  begin  FRequiredArray[i].Required := Value;  RecRnd := true;  break;  end;  end;  if not RecRnd then  begin  SetLength(FRequiredArray, Length(FRequiredArray) + 1);  FRequiredArray[High(FRequiredArray)].IEN := aIEN;  FRequiredArray[High(FRequiredArray)].Required := Value;  end;  end; |

Create

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | Create | | | |
| **Short Description** | Constructor | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| constructor TGMV\_TemplateVital.Create;  begin  inherited;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| constructor TGMV\_TemplateVital.Create;  begin  inherited;  SetLength(FRequiredArray, 0);  end; |

CreateFromXPAR

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | CreateFromXPAR | | | |
| **Short Description** | Create the vital form from server settings | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: XPARVal | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| constructor TGMV\_TemplateVital.CreateFromXPAR(XPARVal: string);  begin  inherited Create;  FIEN := Piece(XPARVal, ':', 1);  try  FVitalName := GMVTypes.Entries[GMVTypes.IndexOfIEN(FIEN)];  except  FVitalName := 'Unknown (' + FIEN + ')';  end;  FMetric := (Piece(XPARVal, ':', 2) = '1');  FQualifiers := Piece(XPARVal, ':', 3);  end; |

| Modified Logic (Changes are in bold) |
| --- |
| constructor TGMV\_TemplateVital.CreateFromXPAR(XPARVal: string);  begin  inherited Create;  FIEN := Piece(XPARVal, ':', 1);  try  FVitalName := GMVTypes.Entries[GMVTypes.IndexOfIEN(FIEN)];  except  FVitalName := 'Unknown ('+FIEN+')';  end;  FMetric := (Piece(XPARVal, ':', 2) = '1');  FQualifiers := Piece(XPARVal, ':', 3);  **RequiredAsString := Piece(XPARVal, ':', 4);**  end; |

Destroy

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | Destroy | | | |
| **Short Description** | Destructor | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| // destructor TGMV\_TemplateVital.Destroy;  // begin  // inherited;  // end; |

| Modified Logic (Changes are in bold) |
| --- |
| destructor TGMV\_TemplateVital.Destroy;  begin  **SetLength(FRequiredArray, 0);**  inherited;  end; |

Vitals Utils

uGMV\_Common

Summary of changes

Added new common functions

Detailed Changes

Uses Clause

Added System.Character to second uses section.

MixedCase

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | MixedCase | | | |
| **Short Description** | Returns a stirng in mix case formating | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: x | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| function MixedCase(const x: string): string;  var  i: integer;  begin  {  NOTICE: IsUpper, ToLower, ToUpper, etc. have been deprecated in XE8 ONLY.  TCharHelper.Methods are the replacements. Merges from XE3 must be carefully  handled here or the XE8 code throws warnings.  }  Result := x;  for i := 2 to Length(x) do  if (not CharInSet(x[i - 1], [' ', ',', '-', '.', '/', '^', '[', ''''])) and x[i].IsUpper then  begin  Result[i] := x[i].ToLower;  end  else if (CharInSet(x[i - 1], [' ', ',', '-', '.', '/', '^', '[', ''''])) and x[i].IsLower then  begin  Result[i] := x[i].ToUpper;  end  else if (x[i] = 'S') and (x[i - 1] = '''') and x[i - 2].IsLetter then  begin  if (i < Length(x)) and (CharInSet(x[i + 1], [' ', ',', '-', '.', '/', '^', '[', ''''])) then  begin  Result[i] := x[i].ToLower;  end  else if (i = Length(x)) then  begin  Result[i] := x[i].ToLower;  end  end;  end; |

PositionForm

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | PositionForm | | | |
| **Short Description** | Sets the postion of the form | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: x | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| procedure PositionForm(aForm: TForm);  var  iHeight,  iWidth: Integer;  begin  iHeight := Screen.WorkAreaHeight;  iWidth := Screen.WorkAreaWidth;  if aForm.Width > iWidth then aForm.Width := iWidth;  if aForm.Height > iHeight then aForm.Height := iHeight;  if aForm.Top> iHeight then aForm.Top := 0;  if aForm.Left> iWidth then aForm.Left := 0;  if aForm.Top < 0 then aForm.Top := 0;  if aForm.Left < 0 then aForm.Left := 0;  if aForm.Top + aForm.Height > iHeight then  aForm.Top := iHeight - aForm.Height;  if aForm.Left + aForm.Width > iWidth then  aForm.Left := iWidth - aForm.Width;  end; |

| Modified Logic (Changes are in bold) |
| --- |
| procedure PositionForm(aForm: TForm);  var  iHeight,  iWidth: Integer;  begin  **if Screen.MonitorCount = 1 then**  **begin**  iHeight := Screen.WorkAreaHeight;  iWidth := Screen.WorkAreaWidth;  **end else begin**  **iHeight := Screen.DesktopHeight;**  **iWidth := Screen.DesktopWidth;**  **end;**  if aForm.Width > iWidth then aForm.Width := iWidth;  if aForm.Height > iHeight then aForm.Height := iHeight;  if aForm.Top> iHeight then aForm.Top := 0;  if aForm.Left> iWidth then aForm.Left := 0;  if aForm.Top < 0 then aForm.Top := 0;  if aForm.Left < 0 then aForm.Left := 0;  if aForm.Top + aForm.Height > iHeight then  aForm.Top := iHeight - aForm.Height;  if aForm.Left + aForm.Width > iWidth then  aForm.Left := iWidth - aForm.Width;  end; |

uGMV\_Engine

Summary of changes

Communtion module for Vitals

Detailed Changes

getSystemParameterListByName

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | getSystemParameterListByName | | | |
| **Short Description** | Return a list from a VistA parameter | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aName | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: Result | | | | |
| Definition: TStringList | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| function getSystemParameterListByName(aName: string): TStringList;  var  SL: TStringList;  begin  SL := TStringList.Create;  CallRPC(RPC\_PARAMETER, ['GETLST', 'SYS', aName], nil, [rpcSilent, rpcNoResChk], SL);  Result := SL;  end; |

getQualifierReqired

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | getQualifierReqired | | | |
| **Short Description** | Load required parameter into name value seperators | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: Result | | | | |
| Definition: TStringList | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| function getQualifierReqired: TStringList;  var  SL: TStringList;  begin  SL := TStringList.Create;  CallRPC(RPC\_PARAMETER, ['GETLST', 'SYS', GMV\_QualifierRequired], nil, [rpcSilent, rpcNoResChk], SL);  Result := SL;  //Convert to name=value stringlist  Result.NameValueSeparator := '^';  // Result.text := StringReplace(Result.text,'^','=', [rfReplaceAll]);  end; |

IsQualifierGlobalRequired

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | IsQualifierGlobalRequired | | | |
| **Short Description** | Checks if a vital is system wide required | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aVital | | | | |
| Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: Result | | | | |
| Definition: Boolean | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| Function IsQualifierGlobalRequired(aVital: String): Boolean;  //var  // SL: TStringList;  begin  result := false;  // inital fill of the list  if not Assigned(FReqQual) then  FReqQual := getQualifierReqired;  // If the lookup does not exist then re fill from the server  { if FReqQual.IndexOfName(aVital) = -1 then  begin  //If list already exist free it since we will be recreating it (should always be assigned but better safe than sorry)  if Assigned(FReqQual) then  FreeAndNil(FReqQual);  //Reload the list  FReqQual := getQualifierReqired;  end; }  //If in the list use the return  if FReqQual.IndexOfName(aVital) > 0 then  Result := FReqQual.Values[aVital] = 'True';  end; |

UpdateQualifierGlobalRequired

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | UpdateQualifierGlobalRequired | | | |
| **Short Description** | Update the internal vital required list | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aVital | | | | |
| Definition: String  Name: aValue  Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: Result | | | | |
| Definition: Boolean | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| Function UpdateQualifierGlobalRequired(aVital, aValue: String): Boolean;  begin  try  FReqQual.Values[aVital] := aValue;  Result := true;  except  result := false;  end;  end; |

setSystemParameter

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | setSystemParameter | | | |
| **Short Description** | Updates a VistA parameter | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aName | | | | |
| Definition: String  Name: aInstance  Definition: String  Name: aValue  Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| function setSystemParameter(aName, aValue, anOption: string): string;  begin  CallRPC(RPC\_PARAMETER, ['SETPAR', 'SYS', aName, aValue, anOption], nil, []);  Result := RPCBroker.Results[0];  end; |

| Modified Logic (Changes are in bold) |
| --- |
| function setSystemParameter(aName, **aInstance, aValue**: string): string;  begin  CallRPC(RPC\_PARAMETER, ['SETPAR', 'SYS', aName, **aInstance**, **aValue**], nil, []);  Result := RPCBroker.Results[0];  end; |

setSystemParameterList

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | setSystemParameterList | | | |
| **Short Description** | Updates a VistA parameter as a list | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: aName | | | | |
| Definition: String  Name: aInstance  Definition: String  Name: aValue  Definition: String | | | | |
| **Output Attribute Name and Definition** | Name: Result | | | | |
| Definition: String | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| function setSystemParameterList(aName, aInstance: string; aValue: TStringList): string;  begin  CallRPC(RPC\_PARAMETER, ['SETLST', 'SYS', aName, aInstance], aValue, []);  Result := RPCBroker.Results[0];  end; |

Finalization

Table 48: Function

| Function | Activities | | | |
| --- | --- | --- | --- | --- |
| **Function Name** | Finalization | | | |
| **Short Description** | Clean up | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
| NA |

| Modified Logic (Changes are in bold) |
| --- |
| Finalization  if assigned(FReqQual) then  FreeAndNil(FReqQual); |

Not the end of the work

#### Design Element Tables

The design element tables are provided for your convenience. Copy each table as many times as necessary to address multiple items within each section. Add rows and headings to the tables to provide any additional required information to define the item or to specify the modifications to the item. Numbering of the design element tables to align them underneath the applicable requirement or sub-requirement is recommended, but is left to the author’s discretion. For that reason they are not numbered in this template.

##### Routines (Entry Points)

This section is an illustration that is VistA specific. The authors are free to organize this information by technology, different templates, or optional sections depending on the task at hand.

Complete the table for each routine affected by the functionality being designed.

Table 14: Routines (Instructions)

| Routines | Instructions |
| --- | --- |
| **Routine Name** | List the routine affected by the functionality being designed. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **RTM** | List the RSD item number within the SDD (i.e., If the RSD has a requirement of 3.3.1, add Support for a new API, then in this column list RSD Requirement 3.3.1) |
| **Related Options** | List options that directly call or are called by the routine. |
| **Related Routines** | List routines that directly call or are called by the routine. |
| **Data Dictionary (DD) References** | List files that reference the routine through input transforms, cross reference logic, etc. |
| **Related Protocols** | List protocols that reference or are referenced by the routine. |
| **Related Integration Control Registrations (ICRs)** | List proposed new ICRs and subscribed ICRs. Also, list any obscure Supported ICRs. |
| **Data Passing** | Check the appropriate box. Also a short description of what invokes the new/changed routine should be included in this section. An example of such a description would be a note that the new/changed routine will be invoked as part of a function call or it would be invoked through user menu-driven options, system protocols, HL7 Logical Links, etc. This section refers specifically to the change implemented with the design. |
| **Input Attribute Name and Definition** | List the Input Attributes passed into the new or changed routine logic. Each attribute should be defined. |
| **Output Attribute Name and Definition** | List the Output Attributes returned from the new or changed routine logic. Each attribute should be defined. |
| **Current Logic** | Define the current logic in the routine that the design will modify. If this is new code, enter “N/A”. |
| **Modified Logic (Changes are in bold)** | Define the logic in the routine that the design will implement. |

Table 15 (Grouping): Routines

| Routines | Activities | | | |
| --- | --- | --- | --- | --- |
| **Routine Name** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** |  | | | |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Routines | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output Reference | Both | Global Reference | Local |
| **Input Attribute Name and Definition** | Name:  Definition: | | | | |
| **Output Attribute Name and Definition** | Name:  Definition: | | | | |

| Current Logic |
| --- |
|  |

| Modified Logic (Changes are in bold) |
| --- |
|  |

##### Templates

Complete Table 16 for each template affected by the functionality being designed. A short description of what change will be made to the templates should be included in this section.

Note: If preferred, copy and paste this section directly from VA FileMan DDs instead of using the tables.

Table 16: Templates (Instructions)

| Templates | Instructions |
| --- | --- |
| **Template Name** | Identify the template affected by the functionality being designed |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **RSD Traceability** | List the Requirement Specification Document (RSD) item number within the SDD (i.e., If the RSD has a requirement of 3.3.1, add Support for a new API, then this column should list RSD Requirement 3.3.1) |
| **Template Type** | Indicate the type of template identified (Sort, Input, or Print). |
| **Related Options** | List options that directly call or are called by the template. |
| **Related Routines** | List routines that directly call or are called by the template. |
| **Data Dictionary (DD) References** | List files/fields that reference the template(s) through input transforms, and cross reference logic. |
| **Global References** | List the ICRs for global references that are outside your namespace. |

Table 17: Templates

| Templates | Description | | | |
| --- | --- | --- | --- | --- |
| **Template Name** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RSD** |  | | | |
| **Template Type** | Sort | Input | Print | Other |
| **Related Options** |  | | | |

| **Related Routines** | **Routines “Called By”** | **Routines “Called”** |
| --- | --- | --- |
|  |  |  |

| Routines | Description |
| --- | --- |
| **Data Dictionary (DD) References** |  |
| **Global References** |  |

##### Bulletins

If the project develops or affects bulletins, then complete this section; if not then state that the section is not applicable and delete the tables and content of the section. Complete the table for each bulletin affected by the functionality being designed. A short description of what change will be made to the bulletins should be included in this section.

Note: If preferred, copy and paste this section directly from VA FileMan DDs instead of using the tables.

Table 18: Bulletins (Instructions)

|  |  |
| --- | --- |
| Bulletins | Instructions |
| **Bulletin Name** | List the specific bulletin affected by the functionality being designed. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **RTM** | List the RSD item number within the SDD (i.e., If the RSD has a requirement of 3.3.1, add Support for a new API, then in this column list RSD Requirement 3.3.1). |
| **Related Options** | List options that directly send the bulletin. |
| **Related Routines** | List routines that directly send the bulletin. |
| **Mail Subject** | List the subject of the mail message, i.e., which bulletin this affects. |
| **Mail Group** | List the mail group (recipients) of the mail message. |
| **Parameters** | List necessary parameters. |
| **Data Dictionary (DD) References** | List files/fields that reference the bulletin(s) through input transforms, cross reference logic, etc. should be listed under Data Dictionary (DD) References. |

Table 19: Bulletins

| Bulletins | Description | | | |
| --- | --- | --- | --- | --- |
| **Bulletin Name** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **RTM** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Routines | Description |
| --- | --- |
| **Mail Subject** |  |
| **Mail Group** |  |
| **Parameters** |  |
| **Data Dictionary (DD) References** |  |

##### Data Entries Affected by the Design

Provide the following data for each field to be created, modified, or deleted or provide a “Before and After: Data Entries Affected by the Design.”

Identify the entries affected by the design. If a blanket change will be made to each entry affected, that change should be defined in this table.

Only changes that are unique to each record should be defined in the Unique Record(s) section (Section 6.2.2.3.5). Redundant information should not be entered into each chart in the Unique Record(s) section.

Table 20: Data Entries Affected by the Design

| Field Name | Current Value | New Value |
| --- | --- | --- |
|  |  |  |

##### Unique Record(s)

List the unique record ID(s) that will be affected by the changes implemented by the design. This is commonly done in the .01 field. The values defined in the Current Value and New Value columns should be the exact value of the data. For each unique record ID, copy this table and provide the information.

Table 21: Unique Record ID

| Field Name(s) | Current Value | New Value |
| --- | --- | --- |
|  |  |  |

##### File or Global Size Changes

Indicate the change to the size of the file or global as a result of the design implemented with this description. Global size changes tie back to the business requirements and RSD. Growth or reduction in the size of the global should be indicated in this section. If the file is static across all VistA systems, a blanket statement of how the change will affect the size of the global will suffice.

For example, “The National Procedure file is a new file and will require 8.7K of disk space to install.”

If a file is dynamic and its size may vary from VistA system to VistA system, the description should indicate the change in the file per record and the number of records that the site may anticipate. For example, if a field is being added to the patient file that will result in an increase of 7K per patient, the site can estimate the global growth based on the number of entries in that file.

Note: If the Capacity Planning analysis is available, then enter it here. If not, then use the Project Team projection.

Table 22: File or Global Size Changes

| File/Global Name(s) | Estimated Increase | Estimated Decrease |
| --- | --- | --- |
|  |  |  |

##### Mail Groups

Complete the table for each of the mail groups affected by the functionality being designed. A short description of what changes will be made to the affected mail groups should be included in this section.

Note: If preferred, this can be captured directly from VA FileMan DDs after the fact.

Table 23: Mail Groups (Instructions)

| Mail Groups | Instructions |
| --- | --- |
| **Mail Group Name** | List the name of the mail group being modified. The mail group name may include a domain name. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **Related Options** | List options that directly reference the file. |
| **Related Routines** | List routines that reference the mail group. |
| **Data Dictionary (DDs) References** | List files that reference the mail group through input transforms, cross-reference logic, etc. |
| **Related Protocols** | List protocols that directly reference the mail group. |
| **Mail Group Description** | Describe the purpose for the mail group. |
| **Self-Enrollment Allowed** | Check the appropriate box either Yes or No. |
| **Type** | Check the appropriate box either Public or Private. |

Table 24: Mail Groups

| Mail Groups | Activities | | | |
| --- | --- | --- | --- | --- |
| **Mail Group Name** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Mail Groups | Instructions | |
| --- | --- | --- |
| **Data Dictionary (DD) References** |  | |
| **Related Protocols** |  | |
| **Mail Group Description** |  | |
| **Self-Enrollment Allowed** | Yes | No |
| **Type** | Public | Private |

##### Security Keys

This section lists the specific security keys affected by the functionality being designed. A short description of the changes that will be made to the security keys affected should be included in this section.

Note: If preferred, this can be captured directly from VA FileMan DDs after the fact.

Table 25: Security Keys (Instructions)

| Security Keys | Instructions |
| --- | --- |
| **Security Key Name** | List the specific name of the security key being modified. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **Related Options** | List options that directly reference the security key. |
| **Related Routines** | List routines that reference the security key. |
| **Data Passing** | Check the appropriate box. Enter a short description of an event that would trigger the new/changed routine, for example, a note that the change to the security key will be referenced through user menu driven options, routines, etc. This section refers specifically to the change implemented with the design. |
| **Security Key Description** | List a brief description of the security key. |
| **Subordinate Keys** | List any subordinate keys. |
| **Mutually Exclusive Keys** | Enter the name of a key that may not be held jointly with this one. |
| **Granting Condition Logic** | Define the logic for the Granting Condition of the Security Key affected by the functionality being designed. |
| **Current Logic** | If the security key currently has a granting condition, define the current logic for that granting condition. If the security key did not exist before, indicate that there is currently no security key. |
| **Modified Logic  (Changes are in bold)** | Define the granting condition that the design will implement. If the security key is new to the field, define the logic here. |
| **Hierarchical Precedence** | Define which key is used if one key will take precedence over another key. |

Table 26: Security Keys

| Security Keys | Activities | | | |
| --- | --- | --- | --- | --- |
| **Security Key Name** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Security Keys | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Security Key Description** |  | | | | |
| **Subordinate Keys** |  | | | | |
| **Mutually Exclusive Keys** |  | | | | |
| **Granting Condition Logic** |  | | | | |

| Current Logic |
| --- |
|  |

| Modified Logic (Changes are in bold) |
| --- |
|  |

| Security Keys | Activities |
| --- | --- |
| **Hierarchical Precedence** |  |

##### Options

Complete the table for each of the options affected by the functionality being designed. A short description of the changes that will be made to the options affected should be included. Changes to the OPTION file (#19) are to be included, not the functionality of the option invoked.

Note: If preferred, this can be captured directly from VA FileMan DD after the fact.

Table 27: Options (Instructions)

| Options | Instructions |
| --- | --- |
| **Option Name**  **(MENU TEXT field)** | Enter the name of the option affected. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change |
| **Associated Menu Options that will invoke this reference** | List the menu type options on which the respective option is or will be contained. |
| **Data Passing** | Check the appropriate box. Also a short description of what invokes the new/changed routine should be included in this section. An example of such a description would be a note that the change to the option will be referenced through VA Mailman server messages, user selection of the option from the VA Kernel Menu Management system, etc. This section refers specifically to the change implemented with the design. |
| **Menu Text Description** | Enter the name of the option as it will be displayed to the user within the menu system. |
| **Option Type** | Specify the type of option |
| **Option Definition** | Provide all the information necessary to fully define the option. Include options that are included in the menu, if applicable. |
| **Current Entry Action Logic** | Define the current logic for the entry action of the option affected by the functionality being designed. If the entry action did not exist before, indicate that there currently is no entry action. |
| **Modified Entry Action Logic (Changes are in bold)** | Define the entry action that the design will implement. If the entry action is new to the field, define the logic here. |
| **Current Exit Action Logic** | Define the current logic for the exit action of the option affected by the functionality being designed. If the exit action did not exist before, indicate that there currently is no exit action. |
| **Modified Exit Action Logic**  **(Changes are in bold)** | Define the exit action that the design will implement. If the exit action is new to the field, define the logic here. |

Table28: Options

| Options | Activities | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Option Name** |  | | | | | | | | | | |
| **Enhancement Category** | New | Modify | | | | Delete | | | No Change | | |
| **Associated Menu Options that will invoke this reference** |  | | | | | | | | | | |
| **Data Passing** | Input | | Output | | Both | | | Global Reference | | | Local Reference |
| **Menu Text Description** |  | | | | | | | | | | |
| **Option Type** | Edit | | | Print | | | Menu | | | Inquire | |
| Action | | | Run Routine | | | Other | | |  | |
| **Associated Routine** |  | | | | | | | | | | |
| **Option Definition** |  | | | | | | | | | | |

| Current Entry Action Logic |
| --- |
|  |

| Modified Entry Action Logic (Changes are in bold) |
| --- |
|  |

| Current Exit Action Logic |
| --- |
|  |

| Modified Exit Action Logic (Changes are in bold) |
| --- |
|  |

##### Protocols

Complete the table for each of the protocols affected by the functionality being designed. A short description of the changes that will be made to the protocols affected should be included in this section. Changes to the PROTOCOL file (#101) are to be included, not the functionality of the protocol invoked.

Note: If preferred, this can be captured directly from VA FileMan DDs after the fact.

Table29: Protocols (Instructions)

| Protocols | Instructions |
| --- | --- |
| **Protocol Name** | List the name of the protocol affected. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **Associated Protocols** | List the ancestors of the protocol being designed, i.e., those protocols that contain the respective protocol as an item. |
| **Data Passing** | Check the appropriate box. An event that would trigger the new/changed protocol should be included in this section. An example would be a note that the change to the protocol will be referenced through the VA event driver, List Manager, user selection of a protocol from the VA Kernel Menu Management system. This section refers specifically to the change implemented with the design. |
| **Item Text Description** | Enter the protocol's text as it appears to the user on the menu or sub-header. |
| **Protocol Type** | Define the type of protocol to be executed |
| **Associated Routine** | List any associated routines affected by the protocol being designed. |
| **Current Entry Action Logic** | Define the current logic for the entry action of the protocol affected by the functionality being designed. If the entry action did not exist before, indicate that there currently is no entry action. |
| **Modified Entry Action Logic  (Changes are in bold)** | Define the entry action that the design will implement. If the entry action is new to the field, define the logic here. |
| **Current Exit Action Logic** | Define the current logic for the exit action of the protocol affected by the functionality being designed. If the exit action did not exist before, indicate that there currently is no exit action. |
| **Modified Exit Action Logic  (Changes are in bold)** | Define the exit action that the design will implement. If the exit action is new to the field, define the logic here. |

Table 30: Protocols

| Protocols | Activities |
| --- | --- |
| **Protocol Name** |  |
| **Enhancement Category** | New  Modify  Delete  No Change |
| **Associated Protocols** |  |
| **Data Passing** | Input  Output  Both  Global Reference  Local Reference |
| **Item Text Description** |  |
| **Protocol Type** | Action  Menu  Protocol  Protocol Menu  Limited Protocol  Extended Action  Dialog  Other |
| **Associated Routine** |  |

| Current Entry Action Logic |
| --- |
|  |

| Modified Entry Action Logic (Changes are in bold) |
| --- |
|  |

| Current Exit Action Logic |
| --- |
|  |

| Modified Exit Action Logic (Changes are in bold) |
| --- |
|  |

##### Remote Procedure Call (RPC)

Complete the table for each RPC affected by the functionality being designed.

Note: If preferred, this can be captured directly from VA FileMan DDs after the fact.

Table 31: RPCs (Instructions)

| RPCs | Instructions |
| --- | --- |
| **Name** | List the specific name of the RPC affected. |
| **TAG^RTN** | List the tag (label) and routine. |
| **Input Parameters** | This field is used to identify an input parameter for the API. |
| **Results Array** | This field tells the RPC Broker how to process the resulting data from the call. |
| **Description** | Provide a brief description of the RPC affected. |

Table 32: RPCs

| RPCs | Activities | | |
| --- | --- | --- | --- |
| **Name** |  | | |
| **TAG^RTN** |  | | |
| **Input Parameters** |  | | |
| **Results Array** | Single Value | Array | Word Processing |
| Global Array | Global Instance |  |
| **Description** |  | | |

##### Constants Defined in Interface

Provide the name and description.

Table 33: Constants Defined in Interface

| Name | Description |
| --- | --- |
|  |  |

##### Variables Defined in Interface

Provide the name, type, and description.

Table 34: Variables Defined in Interface

| Name | Type | Description |
| --- | --- | --- |
|  |  |  |

##### Types Defined in Interface

Provide the name, type, and description.

Table 35: Types Defined in Interface

| Name | Type | Description |
| --- | --- | --- |
|  |  |  |

##### GUI

List the GUI affected by the functionality being designed and include a short description of the changes made to the affected GUI. The headers in the following tables have names for the information outlined. There are a number of items in this section that would generally be global information and visible to all other aspects.

Table 36: GUI

| Unit Name | Description |
| --- | --- |
|  |  |

##### GUI Classes

Table 37: GUI Classes (Instructions)

| GUI Classes | Instructions |
| --- | --- |
| **Class Name** | List the name of the class affected. The headers in the following tables have names for the information outlined. Note that only the new properties and methods for a class are listed below. All ancestor properties and methods are still available and unchanged. |
| **Derived From Class** | List the class that this is derived from, its parent and any interfaces listed as part of this class. |
| **Purpose** | Describe the functionality that users can access from this class and related form, if any. |

Table 38: GUI Classes

| GUI Classes | Instructions |
| --- | --- |
| **Class Name** |  |
| **Derived From Class** |  |
| **Purpose** |  |

##### Current Form

Provide a screen capture or graphical representation of the current layout.

##### Modified Form

Provide a screen capture or graphical representation of the layout that the design will implement.

##### Components on Form

Table 39: Components on Form

| Name | Type | Description |
| --- | --- | --- |
|  |  |  |

##### Events

Table 40: Events

| Name | Type | Description |
| --- | --- | --- |
|  |  |  |

##### Methods

Table 41: Methods

| Method Name | Procedure/Function | Description |
| --- | --- | --- |
|  |  |  |

##### Special References

Include references that are not listed elsewhere.

| Special Reference Name | Type | Description |
| --- | --- | --- |
|  |  |  |

##### Class Events

Table 42: Class Events

| Name | Type | Description |
| --- | --- | --- |
|  |  |  |

##### Class Methods

Table 43: Class Methods

| Name | Procedure/Function | Description |
| --- | --- | --- |
|  |  |  |

##### Class Properties

Table 44: Class Properties

| Class Properties Name | Type | Visibility | Description |
| --- | --- | --- | --- |
|  |  |  |  |

##### Uses Clause

Use this section to provide a uses clause that lists the other units (code or form units) that this unit will use. This may be documented in the form of a Unified Modeling Language (UML) drawing.

##### Forms

This section lists the forms that will be affected or created by the functionality being designed. A short description of the change that will be made to the forms should be included.

Table 45: Forms (Instructions)

| Forms | Instructions |
| --- | --- |
| **Form Name** | List the name of the form affected by the functionality being designed. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **Form Functionality** | Describe the form’s functionality and refer to the usage of the form. An example of such a description is “This form is used to enter patient demographic data.” |
| **Current Form Layout** | Define the current form layout that the design will modify. If this is a new form, enter “N/A”. |
| **Modified Form Layout (Changes are in bold)** | Define the form layout that the design will implement. |

Table 46: Forms

| Forms | Description | | | |
| --- | --- | --- | --- | --- |
| **Form Name** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Form Functionality** |  | | | |

| Current Form Layout |
| --- |
|  |

| Modified Form Layout (Changes are in bold) |
| --- |
|  |

##### Functions

The functions affected by the capabilities being designed should be listed in this section. A short description of what change will be made to the functions and/or new functions should be included.

Table 47: Functions (Instructions)

| Functions | Instructions |
| --- | --- |
| **Function Name** | List the specific function affected by the capability being designed. |
| **Short Description** | List a short description of the change that will be made to the functions and/or new functions. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **Related Options** | List the options that directly call or are called by the function. |
| **Related Routines** | List the routines that directly call or are called by the function. |
| **Data Dictionary (DD) References** | List the files that reference the function through input transforms, cross reference logic, etc. |
| **Related Protocols** | List the protocols that reference or are referenced by the function. |
| **Related Integration Control Registrations (ICRs)** | List proposed new ICRs and subscribed ICRs. Also, list any obscure Supported ICRs. |
| **Data Passing** | Check the appropriate box. An event that would trigger the new/changed function should be included in this section. An example of such a description would be a note that the new/changed function will be invoked as part of a function call or it would be invoked through system protocols, HL7 Logical Links, etc. This section refers specifically to the change implemented with the design. |
| **Input Attribute Name and Definition** | List the input attributes passed into the new or changed function logic. Each attribute should be defined. |
| **Output Attribute Name and Definition** | List the output attributes returned from the new or changed function logic. Each attribute should be defined. |
| **Current Logic** | Define the current logic in the function that the design will modify. If this is new code, enter “N/A”. |
| **Modified Logic (Changes are in bold)** | Define the logic in the function that the design will implement. |

Table 48: Forms

| Function Name | Activities | | | |
| --- | --- | --- | --- | --- |
| **Short Description** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Related Options** |  | | | |

| Related Routines | Routines “Called By” | Routines “Called” |
| --- | --- | --- |
|  |  |  |

| Function Name | Activities | | | | |
| --- | --- | --- | --- | --- | --- |
| **Data Dictionary (DD) References** |  | | | | |
| **Related Protocols** |  | | | | |
| **Related Integration Control Registrations (ICRs)** |  | | | | |
| **Data Passing** | Input | Output | Both | Global Reference | Local Reference |
| **Input Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |
| **Output Attribute Name and Definition** | Name: | | | | |
| Definition: | | | | |

| Current Logic |
| --- |
|  |

| Modified Logic (Changes are in bold) |
| --- |
|  |

##### Dialog

In this section list the changes to the DIALOG file (#.84).

Table 49: Dialog (Instructions)

| Dialog | Instructions |
| --- | --- |
| **Dialog Message (Description)** | List the specific message affected or needed by the changes being designed. |
| **Enhancement Category** | Select the appropriate category: New, Modify, Delete, or No Change. |
| **Dialog Message (Description) Condition** | Describe the dialog message (description) functionality. An example of such a description would be the condition that would trigger the output of the message (dialog). This section refers to the condition generating the message (dialog). |
| **Current Dialog Message (Description)** | Define the current dialog message (description) that the design will modify. If this is a new dialog message (description) enter N/A. |
| **Modified Dialog Message (Description)  (Changes are in bold)** | Define the dialog message (description) that the design will implement. |

Table 50: Dialog

| Dialog | Instructions | | | |
| --- | --- | --- | --- | --- |
| **Dialog Message (Description)** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Dialog Message (Description) Condition** |  | | | |
| **Current Dialog Message (Description)** |  | | | |
| **Modified Dialog Message (Description)  (Changes are in bold)** |  | | | |

##### Help Frame

A short description of what change will be made to the Help Frame text and/or new text should be included in this section. Help frames may be associated with options or with data dictionary fields to provide on-line instruction.

Table 51: Help Frame (Instructions)

| Help Frame | Instructions |
| --- | --- |
| **Help Frame Text** | List the text affected or needed by the changes being designed. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **Help Frame Text Calling Mechanism** | Provide a short description of the mechanism used to call the Help Frame text in this section. An example of a mechanism would be the name of the routine or an explanation of how the Help Frame is called. An example of a calling mechanism would be the Standard VA FileMan API and the keystroke(s) that would trigger the output of the text. |
| **Current Help Frame Text** | List the current Help Frame Text that the design will modify. If new text enter N/A. |
| **Modified Help Frame Text (Changes are in bold)** | List the Help Frame Text that the design will modify. |

Table 52: Help Frame

| Help Frame | Description | | | |
| --- | --- | --- | --- | --- |
| **Help Frame Text** |  | | | |
| **Enhancement Category** | New | Modify | Delete | No Change |
| **Help Frame Text Calling Mechanism** |  | | | |

| Current Help Frame Text |
| --- |
|  |

| Modified Help Frame Text (Changes are in bold) |
| --- |
|  |

##### HL7 Application Parameter

Table 53: HL7 Application Parameter (Instructions)

| HL7 Application Parameter | Instructions |
| --- | --- |
| **HL7 Application Parameter Name** | List the HL7 Application Parameter affected or needed by the changes being designed. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **Application Status** | Check the appropriate box in the applicable column for Current and Modified |
| **Facility Name** | List the current and modified value in the appropriate column. |
| **Country Code** | List the current and modified value in the appropriate column. |
| **HL7 Field Separator** | List the current and modified value in the appropriate column. |
| **HL7 Encoding Characters** | List the current and modified value in the appropriate column. |
| **Mail Group** | List the current and modified value in the appropriate column. |

Table 54: HL7 Application Parameter

| HL7 Application Parameter Name | Description |
| --- | --- |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Enhancement Category** | New | Modify | Delete | No Change |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Application Status** | Active | Inactive | Active | Inactive |

| Enhancement Category | Current | Modified |
| --- | --- | --- |
| **Facility Name** |  |  |
| **Country Code** |  |  |
| **HL7 Field Separator** |  |  |
| **HL7 Encoding Characters** |  |  |
| **Mail Group** |  |  |

##### HL7 Logical Link

Table 55: HL7 Logical Link (Instructions)

| HL7 Logical Link | Instructions |
| --- | --- |
| **HL7 Logical Link Parameter (LLP) Name** | List the specific HL7 Logical Link affected or needed by the changes being designed. |
| **Enhancement Category** | Check the appropriate box: New, Modify, Delete, or No Change. |
| **Node** | List the current and modified value in the appropriate column. |
| **Institution** | List the current and modified value in the appropriate column. |
| **Domain** | List the current and modified value in the appropriate column. |
| **Autostart** | List the current and modified value in the appropriate column. |
| **Queue Size** | List the current and modified value in the appropriate column. |
| **LLP Type** | List the current and modified value in the appropriate column. |

Table 56: HL7 Logical Link

| HL7 Logical Link | Description |
| --- | --- |
| **HL7 Logical Link Parameter Name** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Enhancement Category** | **New** | **Modify** | **Delete** | **No Change** |

| Enhancement Category | Current | Modified |
| --- | --- | --- |
| **Node** |  |  |
| **Institution** |  |  |
| **Domain** |  |  |
| **Autostart** |  |  |
| **Queue Size** |  |  |
| **LLP Type** |  |  |

##### COTS Interface

The specific communication method(s) and Application Interface(s) that will be created or modified for the COTS system being interfaced should be described in this section. A short description of the existing tools that will be used and any new tools that will be developed should also be included.

Table 57: COTS Interface (Instructions)

| COTS Interface | Instructions |
| --- | --- |
| **Communication Method** | List the specific communication method created or modified for the functionality being designed. |
| **Application Interface** | List the specific application interface created or modified for the functionality being designed. |

Table 58: COTS Interface

| COTS Interface | Description |
| --- | --- |
| **Communication Method** |  |
| **Application Interface** |  |

## Network Detailed Design

Provide enough detailed information about the communication requirements to build and/or procure the communication components for the system. This section should provide sufficient detail to support the procurement of hardware for the system installation. Include the following information in the form of detailed designs (as appropriate):

* Details of servers and clients to be included on each area network
* Specifications for bus timing requirements and bus control
* Format(s) for data being exchanged between components
* Diagrams showing connectivity between components, data flow (if applicable), and distances between components
* LAN topology.

## Security and Privacy

### Security

Describe specific security mechanisms at the application level, as guided by NIST 800-53 revision 3 (or most current version). Also, summarize the security mechanisms to be provided by the VA GSSs. Reference the Security Risk Assessment.

The following information should be provided to address security controls:

A high-level description of the security controls, grouped according to the 18 control families identified in NIST 800-53 revision 3 (or most current version). A description of all 18 control families must be addressed; if a control family is not applicable, then state that control family does not apply and explain why it does not apply.

A description of the specific security controls that will be provided by existing VA infrastructure or VA GSSs.

Describe the planned use by the application of the infrastructure’s centralized security mechanisms and VA GSSs (in particular, the identification and authentication, access control, and audit mechanisms), and infrastructure mechanisms, (e.g., Directory Services) to store user account information. Sufficient detail should be provided to show the feasibility of the integration and/or inter operation of application security mechanisms with infrastructure security mechanisms.

### Privacy

Identify privacy design considerations. Describe specific privacy mechanisms at the application. Describe how the application’s privacy requirements will be met. Reference the System Security Plan (SSP) and Privacy Impact Assessment (PIA).

## Service Oriented Architecture / ESS Detailed Design

This section provides details of provided and consumed services as follows:

* Consumed Services: Provide link to Service Description Document for each consumed service.
* Provided Services: Give service design for each provided service.

The information you provide here will be used to upload to the ESS Registry and Repository. At some point in the near future, we do not expect these SOA artifacts such as SLA, Service Description, etc. to be static documents. They will be dynamically generated from the ESS Registry and Repository tool in the form of reports. Any application and service integration design is also documented here.

A list of currently available Enterprise Shared Services is available here: <insert link to ESS list>

### Service Description for <Consumed Service Name>

Provide link to Service Description document for the consumed service. This section will repeat for each consumed service. The Service Description includes Service Interface and Service Level Definition (SLD) to address anticipated capacity requirements.

### Service Design for <Provided Service Name>

This section should describe the detailed service design for each ESS and SOA service needed to obtain an intended result. The Service Design includes Service Interface and Service Level Definition (SLD) to address anticipated capacity requirements.

This section will repeat for each **provided** service.

#### Introduction

##### Purpose and Scope of Service

This service was described at a high level in the charter document. Please refer to it here via a link.

##### Links to Other Documents

Provide links to other documents created for this service so far in the SOA lifecycle. At a minimum, provide links to:

* Service Charter
* Service Roadmap
* Service Description

#### Service Details

##### Service Identification

This section will be written as a table to provide a quick reference to the service's what, where, why and how - cheat sheet.

| Service Attribute | Value |
| --- | --- |
| Name and Alias (if any) | Name of the service and other names for the service, which might be used by someone searching for this service. Please follow ESS naming standards. |
| Overview | Brief textual overview of the service. |
| Version | Version number of the service being described here |
| Latest Status | This field shows the latest status for the above referenced version of this service! The status of a service shows the progress of the service from initiation through development, deployment, and eventual retirement. The status also has a status date associated with the status - and we will be using the latest one here in this document. Valid values include: Inception, Design, Provisioning, Certification / Testing, Operation, Deprecated, Retired, Rejected - Owner has decided not to develop the service. |
| Service Type | Used to define applicable architecture patterns. Examples (from Open Group):  • Interaction  • Process  • Information  • Partner  • Business Application  • Access  • Service Connectivity |
| Architecture Layer | Referred to as class in VA Service template. Used to define applicable architecture patterns and relationships to governing bodies. Examples:  • Solution  • Process  • Information  • Utility  • Underlying |
| Business Domain | Business Vertical or Business Division where this service belongs. |
| Service Domain | The service or technical domain that the service belongs to. Can be used to establish the namespace. |
| Business Organization and Owner | Person who approves this service & any changes. Include email. |
| Technical Organization and Owner | Person responsible for provisioning (specifying, acquiring certifying) this service. Include email. |
| Development Organization and Owner | Person who is responsible for the development processes and activities for this service. Include email. |
| Support Organization and Owner | Person who is responsible for the support of this service while in production. Include email. |
| Target Consumer Organization(s) and Owner(s) | Organizations and/or developers roles that service is intended for. |

##### Service Versions

|  |  |  |
| --- | --- | --- |
| Version Numbers | Current Status of Version | A Brief Description of the change implemented in that version |
| This version | Being Designed |  |
| Example: version 2 | Example: In production. Will be retired with this release. | Example: This release added the ability to look up a person by address.  Provide a link to each version of the service. |
| Example: version 1 | Example: Retired. | Example: This release provided the base minimum functionality to look up a person by name.  Provide a link to each version of the service. |

##### Summary of Design and Platform Details

###### SOA Pattern(s) Implemented

Name of the SOA pattern implemented – for instance, this may be a Pub/Sub model. Just a name and reference to the document or book with the pattern is sufficient for popular patterns or VA's own patterns. If you are using some esoteric pattern, more details will help.

###### COTS Platform vendor names and versions for hosting platform

Example, TIBCO.

#### Dependencies

The Dependency Model identifies other services, systems, databases, etc. that [Service Name] is dependent upon or interacts with to perform its function.

This section should clearly identify all sources and external systems that are accessed by this service to fulfill the service consumers’ request. This section should include diagrams to show as much detail as necessary to inform the developer. Provide a context diagram for the service.

Note: Here our primary audience includes the providers of the service. So this document in general will emphasize system components and sub-systems as much as external interactions.

#### Service Design Details

The next sub-section on Interface Technical Specs **could be** just a copy from the corresponding sub-section in Interface section in the Service Description Document. Here, you could provide more detail necessary for building this service but **the interface spec needs to be consistent between this document and the Service Description Document**. This section contains all information necessary to fully describe an interface published by this service...

##### Interface Technical Specs

The technical specification allows developers of service consumers to locate and discover the service for run time consumption.

###### Service Invocation Type

Such as: SOAP over HTTP, REST.

###### Service Interface Type

Such as: WSDL via Web Service 2.0

###### Service Name

Technical Service Name. Comply with ESS naming standards.

###### Interface

Link to WSDL or other interface document.

###### End Points

Provide if known! Calls that can be made into the service. Can be referenced to the WSDL or can be in a separate table.

###### Operations or Methods

In the table below, the technical names of the operations, inputs and outputs are used. Inputs and outputs, if parameters, must have a data type.

Non-primitive data types must be defined in the Service Information Model section.

This table could be generated automatically from the WSDL content or its equivalent.

Style can take any of these values: Parameters or Document; and One-way or Request-response or Solicit-response or Notification.

Use a separate column for the operation purpose if you wish.

You might use abbreviations in the Faults column and explain the abbreviations used below the table. For example, NF = Not Found, MI = Missing Input.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation Name | Inputs | Outputs | Transactional Qualities if relevant (Updating?, Atomic?, Can participate in transaction?) | Pre and Post Conditions | Exception (s) |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Provide a link to the Service Information model so that the consumer of your system knows the schema for the input and output parameters.

###### Message Schemas

Provide definitions or links to definitions of the message(s) related to the service operations. These may be dependent on the implementation style and protocol binding of the interface.

##### Information Model

Even though this section looks similar to the corresponding section 3.2 in Service Description, remember that the primary objective here is to facilitate construction and to gain approvals from governing bodies. So you will provide more of a “white box” view of the design here to help your developers code the service.

###### Class Diagram and Description of Entities Involved

Map out all entities involved in the service: input, output, exceptions, entities manipulated in persistent media/DBs, intermediate entities created in memory etc.

###### Mappings from ELDM to Standards Based Schemas

Provide mappings from your native schema to any standards based schemas your service will use to communicate outside. For instance, if you are using HL7 based messages then you will show how data is converted from your native schema to HL7.

##### Behavior Model (AKA Use Case Realization)

The Behavior Model defines the actions and processes supported by the service. Actions and methods represented in the use cases and sequence diagrams shown below are further defined by the operation contracts and the message payloads.

###### Use Cases (Use Case Model)

Describe how this service fits into the larger use case model of the consumer. You may need multiple models for multiple consumers. Focus is **not** on the internal workings of the new service instead of the calls made from external consumers. Just a summary or the Use Case Diagram may be sufficient. List the alternative and exception flows. Reference the detailed design documents via a URL.

###### Interaction Diagrams

Cut and paste screen shot from RSA or similar tool or provide link to the model. Provide description to help developers build your service. The interaction diagrams should depict external interactions and internal sequences of calls between internal components. The sequence diagram should cut through all layers to show the main, alternate and exception flows.

#### Gap Analysis

Provide a Gap Analysis (Reference) to demonstrate compliance of this service with various standards, policies, guidelines and laws. The Gap Analysis may take the form of a matrix as shown in the sample below. This will help the governance boards expedite your request.

| Design Elements🡪  Policies / SLD elements etc.↓ | Design  Element A | Design  Element B | Design  Element C | Comment for non-conformance |
| --- | --- | --- | --- | --- |
| Policy X | Match |  |  |  |
| Policy Y |  | Partial |  |  |
| Policy Z |  |  |  | Commercial encryption server in prod will have to address this policy. |
| Policy A |  |  |  | Compliance with this policy not required until next year. |
| New / Additional Features |  |  | New element minimizes manual intervention |  |

##### Variances from Enterprise Target Architecture

This list of “variances” will become a submission to the ESS dispensation process.

##### Variances from SLDs

This list of “variances” will become a submission to the ESS dispensation process.

##### Variances from Standards and Policies

This list of “variances” will become a submission to the ESS dispensation process.

##### Justification for Exceptions and Mitigation

This section will list out any non-functional and functional requirements that are not being met. The non-conformance may be in violation of elements of SLDs, enterprise architecture (TRM Technology Reference Model), privacy policies or guidelines. For each exception provide:

1. Reasons for non-conformance (cost, time, technology, etc.)
2. Mitigating actions taken to reduce the impact of non-conformance
3. Plan (roadmap) to come back into conformance

This list can grow depending on what the Review bodies may ask for.

# External System Interface Design

This section details interfaces external to system, that are NOT services (ESS/SOA). Typically, these may include, RPCs, Flat Data Files etc.

External systems are systems that are not within the scope of the system under development, regardless of whether the other systems are managed by the vendor or its client.

In this section, describe the interface(s) between the system under development (i.e., the system that is the subject of this SDD) and external systems and/or subsystem(s).

It is best to illustrate these sections with annotated diagrams to clearly identify the various elements of the interfaces.

## Interface Architecture

Describe the interface(s) between the system being designed and other systems. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide diagrams showing the communications path(s) between this system and other systems.

## Interface Detailed Design

Provide sufficient detail about the interface requirements for the development team to format, transmit, and/or receive data across the interface.

Include the following information (as appropriate):

* Data format requirements; if data must be reformatted before it is transmitted or after incoming data is received. Describe the tools and/or methods for the reformat process.
* Specifications for hand-shaking protocols between systems; content and format of hand-shake messages, timing for exchanging these messages, and errors handling.
* Format(s) for reports exchanged between the systems.
* Graphical representation of the connectivity between systems, showing the direction of data flow.
* Query and response descriptions.
* Describe the individual data elements that the interfacing entity(s) will provide, store, send, access, and receive, such as:
* Names/identifiers
  + Data Element Name
  + Data Format/Length
  + Data Type
  + Definition
  + Non-Technical Name
  + Non-Technical Synonyms
  + Specifications
  + Synonyms
* Range or enumeration of possible values (e.g., 0-99)
* Accuracy and precision (number of significant digits)
* Priority, timing, frequency, sequencing, and other constraints
* Security and privacy constraints
* Sources (setting/sending entities) and recipients (using/receiving entities).

Describe the data element assemblies (records, messages, files etc.) that the interfacing entity(s) will provide, store, and send, such as:

* Names/identifiers
  + Technical Name, e.g., data structure name
  + Non-technical Names, e.g. synonyms
* Data elements
* Medium/structure of data elements/assemblies
* Visual characteristics (e.g. layouts, fonts, icons etc.)
* Relationships among assemblies
* Security and privacy constraints
* Sources and recipients.

Describe the communication methods that the interfacing entity(s) will use for the interface, such as:

* Communication links, bands, frequencies, and media
* Message formatting
* Flow control (e.g. sequence numbering)
* Data transfer rate
* Routing
* Transmission services
* Safety
* Security and privacy considerations.

Describe characteristics of the protocols that the interfacing entity(s) will use for the interface, such as:

* Priority/layer of the protocol
* Packeting
* Legality checks, error control
* Recovery procedures
* Synchronization
* Status, identification, and other reporting features.

Where appropriate describe other characteristics, such as physical compatibility of the interfacing entity(s) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

# Human-Machine Interface

Describe the human-machine interface (i.e., GUI) relative to the user. Additional information may be added if the suggested headings are inadequate.

## Interface Design Rules

Identify conventions and standards for designing the GUI.

## Inputs

Identify the input media used by the user (i.e., operator) for providing information to the system, such as data entry screens, optical character readers, bar scanners, etc.

Identify the messages associated with operator inputs, including the following:

* Form(s) if the input data is keyed or scanned for data entry
* Access restrictions
* Security considerations.

## Outputs

Describe the system output design relative to the user. System outputs include reports, data display screens, query results, etc.

Identify the following, if appropriate:

* Access restrictions or security considerations
* Description of the purpose of the output
* Report requirements, including frequency of periodic reports
* Screen contents. (Provide a graphic representation of each layout. Define all data elements associated with the layout).

## Navigation Hierarchy

Provide a diagram of the navigation hierarchy that shows how a user moves through the GUI.

### Screen [x.1]

Provide the layout of all input data screens or GUIs. Provide a graphic representation of each GUI, for example, a low-resolution screenshot. Define all data elements associated with each screen or GUI, or reference the data dictionary. Label each data input screen and/or GUI.

### Screen [x.2]

Provide a graphic representation of each GUI, for example, a low-resolution screenshot. Define all data elements associated with each screen or GUI, or reference the data dictionary.

### Screen [x.3]

Provide a graphic representation of each GUI, for example, a low-resolution screenshot. Define all data elements associated with each screen or GUI, or reference the data dictionary.

# Attachment A – Approval Signatures

This section is used to document the approval of the System Design Document. The review should be conducted face to face where signatures can be obtained ‘live’ during the review. If unable to conduct a face-to-face meeting then it should be held via LiveMeeting and concurrence captured during the meeting. The Scribe should add /es/name by each position cited. Example provided below.

The Business Sponsor and Project Manager are required to sign.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

< Business Sponsor >

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

< Project Manager >

1. Additional Information

Attach any addition information that supplements the design specification.

* 1. Identification of Technology and Standards

Identify the system and software which apply to the SDD, including: identification number(s), title(s), abbreviation(s), version number(s), and release number(s). Identify all standards (e.g., American National Standards Institute [ANSI], International Organization for Standardization [ISO], Institute of Electrical and Electronics Engineers [IEEE], etc.).

* 1. Constraining Policies, Directives and Procedures

Identify any constraints or requirements placed on this document by policies, directives, or procedures.

* 1. Requirements Traceability Matrix

Include an RTM that traces modules and data structures to the software requirements. A reference to the location of the RTM is also acceptable.

* 1. Packaging and Installation

Outline any special considerations for software packaging and installation.

* 1. Design Metrics

Describe all metrics to be used during the design activity.

Template Revision History

| Date | Version | Description | Author |
| --- | --- | --- | --- |
| June 2015 | 2.10 | Changed Heading 1 default setting to eliminate page break before | Process Management |
| May 2015 | 2.9 | Edited for Section 508 conformance and remediated with Common Look Office tool | Process Management |
| February 2015 | 2.8 | Incorporates revisions from PMAS Reform Lockdown; namely removing requirements for information that can be obtained from other PMAS authoritative sources. | Andrew Slawter, Office of Technology Strategies |
| September 2014 | 2.7 | Adds Enterprise Shared Services terms and requires AERB Compliance Certificate attachment. | Process Management |
| August 2014 | 2.6 | Signature block update authorized by AERB CR\_018934 | Process Management |
| March 2014 | 2.5 | Section 508 repairs to new version approved by AERB Chair approved | Process Management |
| August 2013 | 2.3 | Replaced the Service Architecture sub-section with new sub-sections for consumed and provided services. Also applied miscellaneous feedback from VA team. | ASD Enterprise Shared Services (ESS) Work Group |
| June 2013 | 1.3 | Upgraded to MS Office 2007-2010 format | Process Management |
| June 2013 | 1.2 | Address inconsistencies in Section 3, Conceptual Design, Correct headings | Process Management |
| March 2013 | 1.1 | Formatted to documentation standards and edited for Section 508 conformance | Process Management |
| January 2013 | 1.0 | Initial Document | PMAS Business Office |

Place latest revisions at top of table.

The Template Revision History pertains only to the format of the template. It does not apply to the content of the document or any changes or updates to the content of the document after distribution.

The Template Revision History can be removed at the discretion of the author of the document.

Remove blank rows.

See TOGAF® 9.1, Part III: ADM Guidelines & Techniques, Gap Analysis on TOGAF website at <http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap27.html>