**Medtronic MITG**

**Research & Development**

**Software Development Plan**

**Gateway Device Management Platform v4.0**

RE00062807 Rev B

|  |  |  |  |
| --- | --- | --- | --- |
| Issue Date: Refer to Agile | | | Page 1 of 35 |
|  | | | |
| Revision History: | | | |
| Revision | SDA | Description | |
| A | RC063055 | Initial Release | |
| B | RC087428 | Document updated to reflect changes in project | |
|  |  |  | |
|  |  | | |
| Author: Refer to Agile for approvals | | | |
| *Ashwini Pandey – PMR R&D* | | | |
|  | | | |
| **Approvals:** Refer to Agile for approvals | | | |
| Maureen Richard – PMR R&D | | | |
| Cliff Schorr- PMR QA | | | |
| *Xishi Pan – Insigma US* [Manual Approval] | | | |
|  | | | |

Table of Contents

[1. INTRODUCTION 5](#_Toc475087741)

[1.1 Purpose 5](#_Toc475087742)

[1.2 Scope 5](#_Toc475087743)

[1.3 System Overview 5](#_Toc475087744)

[1.3.1 Client Application 5](#_Toc475087745)

[1.3.2 Agent 5](#_Toc475087746)

[1.3.3 Server Software 5](#_Toc475087747)

[1.4 Organizational Responsibilities 6](#_Toc475087748)

[1.4.1 Medtronic HI&M Organization 6](#_Toc475087749)

[1.4.2 Medtronic IS Global Shared Services 6](#_Toc475087750)

[1.5 Tailoring of ES-075 6](#_Toc475087751)

[2. REFERENCED DOCUMENTS 6](#_Toc475087752)

[3. Definitions and Acronyms 7](#_Toc475087753)

[4. REGULATORY ASSESSMENT 8](#_Toc475087754)

[4.1 Intended Use Statement 8](#_Toc475087755)

[4.2 FDA Classification 8](#_Toc475087756)

[4.3 IEC 62304 Safety Classification 8](#_Toc475087757)

[5. SOFTWARE DEVELOPMENT 8](#_Toc475087758)

[5.1 Maintenance Lifecycle Phases 8](#_Toc475087759)

[5.1.1 Scheduled Maintenance Release 8](#_Toc475087760)

[5.1.2 Security Patch Release 8](#_Toc475087761)

[5.1.3 Emergency Patch Release 8](#_Toc475087762)

[5.1.4 New Feature Release 9](#_Toc475087763)

[5.1.5 New Device Integration Release 9](#_Toc475087764)

[5.2 Lifecycle Phase Deliverables 9](#_Toc475087765)

[5.3 Software Revision Level History 10](#_Toc475087766)

[5.4 Production Release 10](#_Toc475087767)

[5.5 Software Development Environment 10](#_Toc475087768)

[5.5.1 Server Environments 10](#_Toc475087769)

[5.5.2 Programming Languages 11](#_Toc475087770)

[5.5.3 Coding Standards 11](#_Toc475087771)

[5.5.4 Development Tools 11](#_Toc475087772)

[5.5.5 Software of Unknown Provenance (SOUP) 12](#_Toc475087773)

[6. RISK MANAGEMENT PLAN 14](#_Toc475087774)

[7. SOFTWARE DOCUMENTATION PLAN 15](#_Toc475087775)

[7.1 Style, Format, and Identification of Documentation 15](#_Toc475087776)

[7.1.1 Title Page 15](#_Toc475087777)

[7.1.2 Page Header 15](#_Toc475087778)

[7.1.3 Page Footer 15](#_Toc475087779)

[7.1.4 Revision History 15](#_Toc475087780)

[7.2 Approval of Documentation 15](#_Toc475087781)

[7.3 Storage of Documentation 15](#_Toc475087782)

[7.4 Modification of Documentation 16](#_Toc475087783)

[7.5 Project Documentation 16](#_Toc475087784)

[7.5.1 Software Development Phase Documentation 16](#_Toc475087785)

[7.5.2 Test Phase Documentation 18](#_Toc475087786)

[8. SOFTWARE BUILD PROCESS 19](#_Toc475087787)

[9. SOFTWARE CONFIGURATION MANAGEMENT PLAN 19](#_Toc475087788)

[9.1 Configuration Management Tools 19](#_Toc475087789)

[9.2 Configuration Items 20](#_Toc475087790)

[9.3 Responsibilities 20](#_Toc475087791)

[9.4 Change Control 20](#_Toc475087792)

[9.4.1 Change Control Board (CCB) 20](#_Toc475087793)

[9.4.2 CCB Decision Criteria 20](#_Toc475087794)

[9.5 Software Versions 20](#_Toc475087795)

[9.6 Configuration Identification 21](#_Toc475087796)

[9.6.1 Mandatory Project Baselines 21](#_Toc475087797)

[9.6.2 Optional Project Baselines 21](#_Toc475087798)

[9.7 Configuration Status Accounting 22](#_Toc475087799)

[9.8 Configuration Audit 22](#_Toc475087800)

[10. Branching Strategy 22](#_Toc475087801)

[11. PROBLEM RESOLUTION PLAN 22](#_Toc475087802)

[11.1 Issue Workflow 22](#_Toc475087803)

[11.2 Roles 23](#_Toc475087804)

[11.3 Issue States 23](#_Toc475087805)

[11.4 Issue Transitions 24](#_Toc475087806)

[11.6 Issue Priority Levels 26](#_Toc475087807)

[11.7 Issue Types 26](#_Toc475087808)

[11.8 Software Tool Anomalies 26](#_Toc475087809)

[11.9 Trend Analysis 26](#_Toc475087810)

[12. QUALITY PLAN 26](#_Toc475087811)

[12.1 Test Strategy 27](#_Toc475087812)

[13. RELEASE CRITERIA 27](#_Toc475087813)

[13.1 Criteria for Release to System Verification Testing 27](#_Toc475087814)

[13.2 Criteria for Release to Production 27](#_Toc475087815)

# INTRODUCTION

## Purpose

This document establishes the plan for the development of the Gateway Device Management Platform (GDMP) v4.0, to be released during the Maintenance Phase of the GDMP software lifecycle. The project team will update and revise this plan as needed throughout the v4.0 project. The Software Development Plan (SDP) for the initial production release of the GDMP can be found in [17] R0030805.

## Scope

The scope of this plan is the v4.0 production release of the GDMP. The objective of the project is to optimize the design and architecture of the current system to make it better meet user needs. This release will also include a new feature of the capability to manage device feature licenses.

The project will ensure that compatibility is retained between the GDMP and previously integrated Client Applications and devices.

This scope of this plan covers only the GDMP, and not the Client applications or the devices.

This plan includes the following components for the Gateway Device Management Platform project:

Regulatory Assessment

Software Lifecycle and Deliverables

Risk Management Plan

Software Documentation Plan

Software Configuration Management Plan

Problem Resolution Plan

Quality Plan

## System Overview

The GDMP v4.0 is a development effort of Health Informatics and Monitoring (HI&M) belonging to PMR. The HI&M team will modify the Agent and Server under this Software Development Plan.

The GDMP is web-based infrastructure providing device management/Device Management capability used by the GBUs to service their devices. It is not a product, and it is not a PDP project. Therefore, it does not fall within the scope of the Change Development Plan (CDP) requirement as specified in [20] QD02871.

### Client Application

The Client Application is a Windows application which interfaces with the device, and with the Agent. It hosts the user interface through which a user updates software on a device. The client applications are developed by individual business units responsible for particular device types. However, the GDMP 4.0 system will also be compatible with Common Client application designed and developed by HI&M group.

### Agent

The Agent is a Windows service which interfaces with the Client Application, and with the Server. It provides services for communicating to the Server on behalf of the device.

### Server Software

The Server Software stores software packages (device images), device manuals, and device configurations.

## Organizational Responsibilities

### Medtronic HI&M Organization

#### HI&M Project Team

The HI&M Project Team is responsible for overseeing the management of the project and the process. The roles assigned to the project are: Project Manager, Product Owner, Enterprise Architect, Application Support Engineer, Quality Analyst (QA) and .UAT Test Engineer.

.

#### Insigma US Development Partner

The Insigma US (IUS) Team is an engineering service provider responsible for design, development and testing of the GDMP components. The IUS Team is responsible for developing GDMP software, writing and executing Unit Tests, performing code reviews, creating and executing System Verification Test Procedures (IQ, OQ), Exploratory Test charters, and End to End Test Procedures. The roles assigned to the project are: Project Manager, Enterprise Architect, Software Development and Test Leads, Software Developers and Testers.

### Medtronic IS Global Shared Services

The IS GSS Team is responsible for providing leadership and guidance for specification and management of the project’s computing resources.

## Tailoring of ES-075

The GDMP is not a medical device and will be developed using [1] ES-075 as a guideline, augmented by [2] SDIM deliverables where necessary. Lifecycle activities will be scaled according to system impact on patient safety, product quality, data integrity, and business risk, as determined by the System Risk Analysis. Because the GDMP is web-based infrastructure, and not a medical device, the project will tailor ES-075 as appropriate to the level of risk.

* The project will deviate from [14] GP-003 in that test data entered into an electronic system will not be printed and signed. Instead, a signature page covering the execution of the test procedure and creation of the test data will be printed, signed, scanned and combined with the electronic test data.
* The project will deviate from [16] GP-060 in that Operations Engineering is not required to approve test documentation. The reason for the deviation is that the GDMP is not a manufactured product.
* The project will deviate from [20] QD02871 in that it will not require a CDP Change Request. The reason is that the GDMP is web-based infrastructure providing Device Management capability used by the GBUs to service their devices. It is not a product, and not a PDP project. Therefore, it does not fall within the scope of the CDP.
* The project will not require a Software Certification Statement because it is not a medical device.
* The project will not produce a Unit Test Report, and Code Review Report as these are not required for IEC 62304 Safety Class A software [4].

# REFERENCED DOCUMENTS

[1] EbD SOP, ES-075 Software Development Procedure

[2] GS10005918 Medtronic Corporate SDIM (Software Development Implementation and Maintenance Methodology)

[3] CFR – Code of Federal Regulations Title 21 Part 11 Electronic Records; Electronic Signatures

[4] IEC 62304:2006 Medical device software – Software life cycle processes

[5] RL-006 Medtronic EbD Safety Risk Analysis

[6] R0030804, Medtronic Gateway Build Process

[7] EBD SOP, ES-080 Release of Programmable Devices

[8] R0030808, Medtronic Gateway Coding Standards

[9] Gateway Project RACI Approval Matrix

[10] EbD SOP, ES-081 Incorporating Usability/Human Factors Engineering into Product Development

[11] R0030938, Gateway Device Management Platform Use Case Diagrams

[12] QD02942, Hazard Analysis for Software Template

[13] T-059, Risk Management Report Template

[14] SSG SOP, GP-003 General Rules

[15] R0047079 GRSP Change Management Gateway

[16] EbD SOP, GP-060 Process Validation, Equipment Qualification and Documentation

[17] R0030805, Software Development Plan, Gateway Device Management Platform

[18] RL-010, Validation of Automated Process Equipment and Quality System Software

[19] QD03461, Software Tool Validation Plan/Report Template

[20] QD02871, Design Control System Post Design Release

[21] QD05005, Risk Management Process

[22] NP046048 Rev E, Software Version Numbering

[23] NP042375 Rev H Non-Product Tools Validation

# Definitions and Acronyms

Baseline – A snapshot of a project configuration at a specific point in time.

CCB – Change Control Board, a group of team members with the appropriate departmental representation, who approve or reject changes that are made to a software baseline.

Jira – The Issue Tracking tool used for the project.

Configuration Item - An entity within a particular configuration that can be uniquely identified at a given reference point. Examples include source code files, requirements and design documents, and verification and validation documents.

GDMP – Gateway Device Management Platform (preferred name for GRSP)

Git – The Configuration Management tool consisting of a file repository and an application used to manage version control of source code and documents, and configuration baselines.

Issue – Issue entered into the Jira Issue Tracking tool. An issue can be a defect, a change request, a new feature, etc.

Issue Tracking System - A computerized database that allows for the electronic creation, storage, editing, and retrieval of software problems and their resolutions.

Release - A specification or product configuration that has been formally reviewed and agreed upon. It can be changed only through formal change control procedures.

Release Candidate - A baseline of code which has been unit and integration tested, and is transferred to the test team for verification as a candidate for release.

Phase Transition - The transition from one software lifecycle phase to the next.

Phase Transition Review - A design review that verifies the outputs for the current software lifecycle phase is complete and approves the transition to the next phase.

PM – Project Manager

Traceability Matrix - A matrix that records the relationship between two or more products of the development process (System requirement or software mitigation to software requirement; software requirement to software design entity; and software requirement to software validation).

SOUP – Software of Unknown Provenance. SOUP refers to any software item that is already developed and generally available and that has not been developed for the purpose of being incorporated into a medical device or software previously developed for which adequate records of the development process are not available.

Version – A revision of a configuration item.

# REGULATORY ASSESSMENT

## Intended Use Statement

The Medtronic Gateway Device Management Device Management Platform, used with a Client Application, is intended to provide qualified Biomedical Technicians, Medtronic Service personnel, Medtronic Sales personnel, distributors, and third party service organizations the ability to manage Medtronic medical devices in a Hospital/Critical Care environment.

## FDA Classification

The GDMP, by itself is not a product, and therefore has no FDA classification. A GBU using the GDMP may classify its product according to its intended use. The GDMP will be developed under the appropriate design controls to support a Medical Device Data System (MDDS) classification, if the intended use of the GBU’s product requires this classification.

## IEC 62304 Safety Classification

The GDMP will be developed under the appropriate design controls to support an IEC 62304 software safety class A (no injury or damage to health is possible) classification.

# SOFTWARE DEVELOPMENT

## Maintenance Lifecycle Phases

The GDMP is a mature system and undergoes various types of maintenance and update activities. The GDMP system can be updated and released as a new version under these five categories:

### Scheduled Maintenance Release

Also known as Patch release, this type of planned release occurs typically twice per calendar year and is meant to fix non-emergency defects found during the course of usage of already released system. This type of release may also include minor system updates to enhance functionality of existing features. The testing of this type of release is limited to bug fix testing targeted to individual bugs and system automated testing to ensure full system functionality. There will be no separate development and test phases for Scheduled Maintenance Release.

### Security Patch Release

Security patch releases fix security vulnerabilities due to environment factors such as in the operating system, database system, or the SOUP components of the system. This release is not meant to fix functional defects of the system so no code changes occur. These updates are done on as needed basis, typically every month or, every other month and automated testing is done to ensure system functionality is not impacted. There will be no separate development and test phases for Scheduled Maintenance Release.

### Emergency Patch Release

Emergency patch release is used when a defect makes the system unusable or drastically curtails system usage. The testing of this type of release is limited to bug fix testing targeted to individual bugs and system automated testing to ensure full system functionality. There will be no separate development and test phases for Scheduled Maintenance Release.

### New Feature Release

New feature release adds new functionality to the existing system thus requiring extensive system level documentation and testing. Each New Feature Release follows documentation scope that is similar to a new system development project. It will typically consist of two sub-phases, the New Feature Development Phase and New Feature Test Phase. The testing will be a combination of focused testing of new feature and automated testing to ensure rest of the system functions as designed.

### New Device Integration Release

The purpose of New Device Integration Release has very limited scope which is to integrate a new device type with the GDMP. It will typically consist of two sub-phases, the New Device Integration Development Phase and the New Device Integration Test Phase. The testing will be a combination of focused testing of the new device integration and automated testing to ensure rest of the system function as designed.

The GDMP v 4.0 is intended to completely redesign the current system as well as to add the feature of license management, thus it will be categorized under New Feature Release under Maintenance phase. The release will consist of two sub-phases- New Feature Development Phase and New Feature Test Phase.

At the completion of each sub-phase, the phase outputs, as well as any approved outputs from previous phases that have been modified, will be accounted for in a Phase Transition Review and documented in a Phase Transition Report. The appropriate baselines will be generated as described in Section 10. All non-compliance issues discovered during the software reviews will be handled as described in Section 11. At each phase transition, the date of transition and the results of the software review(s) shall be documented in the Phase Transition Report. Approval of the Phase Transition Report will constitute formal approval of the baseline and the documented resolutions to the non-compliance issues.

## Lifecycle Phase Deliverables

| Lifecycle Phase | Deliverables | Responsibility |
| --- | --- | --- |
| Maintenance New Feature Development Phase | Software Development Plan (SDP) (this document) | HI&M Project Team |
|  | Updated System Risk Analysis (SRA) | HI&M Project Team |
|  | Updated System Requirements Specification (SRS) | IUS Team |
|  | Updated (or new) Traceability Matrix | IUS Team |
|  | Updated (or new) Architecture Document | IUS Team |
|  | Detailed Design Description | IUS Team |
|  | System Verification Plan | IUS Team |
|  | System Verification Procedures (IQ, OQ) | IUS Team |
|  | End to End Test Procedures | IUS Team |
|  | Message Specifications | HI&M Project Team |
|  | Software Revision Level History | IUS Team |
|  | Source Code | IUS Team |
|  |  |  |
|  | Phase Transition Report (PTR) | HI&M Project Team |
|  |  |  |
| Maintenance New Feature Test Phase | Any Development Phase deliverables that were updated | Depends on the deliverable |
|  | IQ Report | IUS Team |
|  | OQ Report | IUS Team |
|  | End to End Test Report | IUS Team |
|  |  |  |
|  | System Regression Test Plan | IUS Team |
|  | System Regression Test Report | IUS Team |
|  | System Verification Summary Report | IUS Team |
|  | Updated Phase Transition Report (PTR) | HI&M Project Team |
|  | Software Deployment Plan | HI&M Team |
|  | Updated Web UI User Guide, if needed | IUS Team |
|  | OTS Software Validation Plan and Procedures | HI&M Project Team |
|  | OTS Software Validation Reports | HI&M Project Team |
|  | UAT Test Plan and Procedures | HI&M Team |
|  | UAT Test Report | HI&M Team |
|  | Software Change Report | IUS Team |
|  | Post Deployment System Readiness Check Procedures | HI&M Team |
|  | Post Deployment System Readiness Check Report | HI&M Team |

## Software Revision Level History

The Software Revision Level History is a log which documents each Release Candidate delivered by the development team to the test team. A Release Candidate is delivered at the end of each sprint. The log includes the date of the Release Candidate, the version number of each executable, and a brief description of the changes between this Release Candidate and the previous Release Candidate. The Software Revision Level History is stored in the Agile PLM system.

At a minimum, the Software Revision Level History shall include the following information:

Person who created the Release Candidate.

Date the Release Candidate was created.

Baseline identifier

Version Number of each executable file.

SHA hash code of each executable file.

Description of the changes.

## Production Release

After completion of the Maintenance New feature Test Phase, the software may be released for general availability. The activities listed in [15] GRSP Change Management must be performed. The production release process will be performed in accordance with [7] ES-080.

## Software Development Environment

### Server Environments

There are several environments for the server software. The validated server environments are hosted in the Terremark environment. The production server is backed up into the Medtronic Data Center.

|  |  |  |  |
| --- | --- | --- | --- |
| Server Environment | Purpose | Description | Location |
| PROD | Production | Validated system | Cloud |
| QA | Validation of major releases | Validated system | Cloud |
| QAPROD | Validation of patch releases | Validated system | Cloud |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Programming Languages

Languages used will be PHP, Java EE, JavaScript, and HTML.

### Coding Standards

Coding standards for the project are defined in [8] Medtronic Gateway Coding Standards.

### Development Tools

The following software development tools will be used for this project. Software development tools will be archived on Master Storage Media (MSM) and included in the appropriate baselines as documented in the Phase Transition Report. All tools will be validated for their intended use. The validation strategy is described in the OTS Software Validation Plan.

| Development Tool | Manufacturer/Open Source Foundation | Release/Version Information | Purpose |
| --- | --- | --- | --- |
| Git | Apache Software Foundation | 1.7.8-3403.108 | Version control of source code |
| Eclipse | Eclipse Foundation | 4.3.1 | IDE |
| PuTTY | Simon Tatham | 0.62 | Telnet/SSH client used for virtual desktop connection to server |
| HtmlUnit | Mike Bowler | 2.1 | Unit test tool for web pages on server |
| JUnit | Kent Beck, Erich Gamma, David Saff | 4.12 | Unit test tool for client and server |
| Jenkins with Ant, Maven, Log4J | Jenkins.org | 1.4.86 | Build automation tool |
| EMMA | Sourceforge.net | 2 | Code coverage tool |
| Selenium | Seleniumhg.org | 2.28.0 | Test automation tool for web applications |
| JMeter | Apache Software Foundation | 2.8 | Server load and performance test tool |
| Find Bugs | Bill Pugh and David Hovemeyer | 3.0.1 | Static Analysis tool |
| NSIS | SourceForge.net | 2.46 | Generate the Agent/Client bundled installation package |
| Remapper | Internal | 3.2- 4.0 | Remap agent pointing to different servers |
| MySQL Workbench | Oracle | 6 | Used to run sql on MySQL database, including query, update |
| WinScp | SourceForge.net | 4.3.9 | Used by Dev/QA to copy file between machines, normally is the data and message files |
| Launch4J | SourceForge.net | 3.0.2 | Wrapping Java applications distributed as jars in lightweight Windows native executables |
| WireMock | GitHub | 1.4 | Used during the Web Service code and testing |
| powermock | Client | 1.6.2 | Mock package |
| easymock | Client | 3.2 | Mock package |
| Log4j | Server, Client | 1.2.17 | logging library |
| mockito | Server | 1.10.19 | Mock creation Utility |
| Mono | Xamarin, Microsoft (formerly by Novell, originally by Ximian), and the Mono community | 4.4.2 | To run .NET applications across other OS platforms |
| Test Automation Framework | Robot | 2.8.5 | Framework that runs test automation suite |
| Test Automation Scripts | Internal | TBD | Test scripts that are run in automated testing |
| Jira | Atlassian Software Systems | 5.2.11 | Issue tracking |
| Fisheye | TBD | TBD | Code review tool |
| Crucible | TBD | TBD | Code review tool |
| Postman Basic | Postdot Technologies/getpostman.com | 4.6.2 | restful API testing (web Browser plug in -- used with which browsers?) |
| InstallShield | Flexera | 2015 Pro | Installer package. |
| Apache Tomcat | Server | 8 | Web Server for DummyAgent, Open source software implementation of the Java Servlet and JavaServer Pages technologies |
| Java | Server, Client | 8u92 | Java platform |
| Internet Explorer | Microsoft | 11 | Internet Browser |
| M2E | Eclipse.org | 1.7.0 | Integration of Apache Maven into Eclipse IDE |
| Maven-assembly plug-in | Apache.org | 2.6 | Aggregate the project output along with its dependencies, modules, and other files into a single distributable format. Used in deployment |
| Jackson-core | Apache.org | 2.8.0 | Basic JSON streaming API implementation used to parse JSON data in ETL tool |
| Jackson-databind | Apache.org | 2.8.4 | General data-binding functionality for Jackson  Used to handle JSON data in ETL tool |

### Software of Unknown Provenance (SOUP)

The following Software of Unknown Provenance (SOUP) will be used for this project. SOUP will be archived on Master Storage Media (MSM) and included in the appropriate baselines as documented in the Phase Transition Report. All SOUP will be validated according to its intended use. The validation strategy is described in the OTS Software Validation Plan.

During the planning activities for each new release of the system, the development team will check for availability of a new version of each SOUP item used in the system. An assessment will be made of the new version and based on the outcome of this assessment; the new version may be validated and incorporated into the system.

In general, only new SOUP versions that are stable should be incorporated into the system. Generally speaking, nightly builds are not acceptable unless they fix a defect which is required for the system to operate successfully. A common approach is to take the first or second minor release after a general availability release. For instance, if the GA release is version v3.1, then the first version which would be incorporated would be v3.1.1. If a particular SOUP package is noted to be very stable, with long times between releases, then this rule may be excepted for that package. For instance, OpenDJ appears to be a candidate for this exception.

Any assessment which results in a change of the SOUP package configuration for the system, shall result in JIRA tasks to modify the following items:

* SOUP repository
* Build dependencies
* Deployment instructions
* This document

|  |  |  |  |
| --- | --- | --- | --- |
| SOUP Software | Where Used | Release/Version Information | Purpose |
| Apache | Server | 2.4.20 | Web server |
| MySQL Community Version | Server | 5.7 | Database for configuration and service data |
| Redhat Linux Enterprise | Server | 6.7 | Server OS |
| OpenSSL | Client | 1.0.0-20.el6 | Encrypted communications |
| MD5Deep | Server, Client | 4.3 | SHA-2 checksum calculation |
| spring | Server, Client | 4.2.4 | Spring framework |
| spring-security | Server | 4.0.4 | Spring Security framework |
| spring-jdbc | Server | 4.2.5 | Spring JDBC |
| common-lang3 | Server | 3.4 | Spring dependence |
| commons-logging | Server | 1.2 | Spring dependence |
| json-jackson | Server, Client | 2.22.2 | JSon parser |
| mybatis-spring | Server | 1.2.4 | Bridge to use MyBites and Spring |
| mybatis | Server | 3.3.1 | DB/SQL utility |
| mysql-connector | Server | 5.1.38 | MySQL connector |
| h2 | Server, Client | 1.4.191 | H2 Database |
| jersey | Server, Client | 2.22.2 | RESTful Web Services |
| jersey-test-framework | Server, Client | 2.22.2 | RESTful Web Services |
| poi | Server | 3.14 | Java API for Microsoft document |
| common-csv | Server | 1.2 | CSV Utility |
| pdfbox | Server | 1.8.11 | PDF Utility |
| c3p0 | Server | 0.9.5.2 | Augmenting traditional JDBC drivers with JNDI-bindable DataSources |
| ehcache | Server | 2.10.1 | Cache |
| org-json | Server | 20160212 | JSon Utility |
| javax.mail-api | Server | 1.5.5 | Java Mail |
| maven-compiler-plugin | Server | 3.5.1 | Maven compiler plugin |
| Netty | Server, Client | 3.5.2 | NIO client server framework which enables quick and easy development of network applications |
| Procrun | Client | 1.0.10 | Allows Windows users to wrap (mostly) Java applications |
| OpenDJ | Server | 3.0.0 | LDAP Engine |
| commons-collections4 | Server | 4.0 | Java Collection operation |
| commons-io | Server, Client | 2.4 | Apache common IO package |
| commons-codec | Client | 1.4 | Apache common Codec package |
| commons-httpcl ient | Client | 3.0.1 | Apache HTTPClient package |
| slf4j | Client | 1.7.7 | Logging package |
| jaxen | Client | 1.1.3 | Java Xpath engine |
| xml-apis | Client | 1.4.01 | XML API package |
| xercesImpl | Client | 2.11.0 | XML Parser |
| jax-rs | Client | 1.1.1 | Java API for RESTFul Service |
| jettison | Client | 1.1 | XML & JSON Binder |
| proguard | Server,Client | 5.2.1 | Obfuscation |
| Digital Rights Management Tool | Server,Client | ? | Generate, encode, and decode licenses |
| SOUP Software | Where Used | Release/Version Information | Purpose |
| Apache | Server | 2.4.20 | Web server |
| MySQL Community Version | Server | 5.7 | Database for configuration and service data |
| Redhat Linux Enterprise | Server | 6.7 | Server OS |
| OpenSSL | Client | 1.0.0-20.el6 | Encrypted communications |
| MD5Deep | Server, Client | 4.3 | SHA-2 checksum calculation |
| spring | Server, Client | 4.2.4 | Spring framework |
| spring-security | Server | 4.0.4 | Spring Security framework |
| spring-jdbc | Server | 4.2.5 | Spring JDBC |
| common-lang3 | Server | 3.4 | Spring dependence |
| commons-logging | Server | 1.2 | Spring dependence |
| json-jackson | Server, Client | 2.22.2 | JSon parser |
| Bootstrap | Web | 3.3.6 | Bootstrap, A Look&Feel library used for cross  platform and display |
| Angular JS | Web | 2 | Ajax library used for GDMP WebUI |

# RISK MANAGEMENT PLAN

Based on [5] RL-006, the following risk management activities will be performed:

Review and update the Software Hazard Analysis as needed, R0031130 Rev B.

Review and update the Risk Analysis Worksheet as needed, R0059458 Rev B.

Update the Risk Management Report, R0031709 Rev D.

Review and update the Residual Risk Summary as needed, R0032416 Rev B.

Create system or software requirements for all newly identified risk mitigations.

Review and revise the Risk Management deliverables as additional features are added to the system.

# SOFTWARE DOCUMENTATION PLAN

This section of the document will specify requirements for creating and modifying software documentation for the project.

## Style, Format, and Identification of Documentation

All source code will be developed according to the conventions outlined in [8] Medtronic Gateway Coding Standards. The remaining software documents will follow the conventions of this section, which includes the guidelines for style, format, and general instructions for preparing and evaluating documentation. This document can be used as an example when preparing project documentation.

### Title Page

The title page shall be the first page within a document. The company, business unit, document title, revision number, author and author’s department shall be clearly indicated. The author shall consist only of the individual who has made the last change. The revision letter shall be consistent with that listed in the last revision history block. The initial approved revision of a document shall be “A”. An approval signature block shall also be included on the title page with the names of the expected approval signatories.

### Page Header

A Page Header shall be consistent and included on each page of a document, excluding the title page. The header shall include the document title, revision number, issue date, and number of pages. The document title shall be consistent with that specified on the title page. The revision number shall be consistent with that listed in the last revision history block. The format for specifying the date on a document shall be by month-day-year. Pages shall be consecutively numbered in Arabic numerals. If the document is divided into volumes, each such volume may restart the page numbering sequence.

### Page Footer

A Page Footer shall be consistent and included on each page of a document, excluding the title page. The footer shall include the company name and a statement of confidentiality.

### Revision History

The revision history for a document shall be on the first page of the document. If a large number of revisions occur, the revision block may overflow into subsequent pages. It shall contain the Revision letter, and the change description. The initial approved revision of a document shall be “A”. Subsequent revisions will roll the revision letter to the next alpha character. The revision letter shall be rolled prior to signature approval of the revised document. The author shall be a typed name (no signatures).

## Approval of Documentation

All document approvers are listed in the [9] RACI chart on the project’s Pulse Connect collaboration site. Documents shall be approved in the Agile PLM system.

## Storage of Documentation

All approved documents shall be stored and controlled in the Agile PLM system. Documents may also be stored on the project’s Pulse Connect collaboration site for ease of access, but Agile is the official repository for project documentation.

## Modification of Documentation

After initial release of a document, the document may be modified as necessary. The modification may be done either by the original author or another project team member. The previously approved version of the document is retrieved from the Agile PLM system, the revision letter incremented, the date updated, and the revision history block completed for the new version. The document modifications shall be done according to the guidance in this document. Approval and storage of modified documents are done according to the guidance in this document.

## Project Documentation

Detailed descriptions of each document’s contents can be found in [1] ES-075. The intended audience for each document can be found in [9] Gateway RACI chart.

### Software Development Phase Documentation

All Development Phase documentation will be generated or updated from a previous project, as needed during the Maintenance Development Phase and approved prior to the Development Phase transition review. Documents will be updated as required throughout the software lifecycle.

#### Software Development Plan (SDP) (this document)

The Software Development Plan defines the lifecycle to be used for the project, the documentation to be generated, the risk management procedures, configuration management procedures, problem resolution procedures, and quality activities for the project

#### System Risk Analysis (SRA)

The System Risk Analysis shall identify all risks, especially safety hazards, associated with the hardware or software for this project and shall be performed as per [5] RL-006. It identifies potential harms resulting from malfunction of the software. It identifies mitigations for these risks, including those mitigations specifically implemented in the software. It is used to provide inputs for developing the system and software safety requirements.

#### User Requirements Document (URD)

The URD documents the user or customer requirements for the system. It is used to provide inputs for developing the system requirements.

#### System Requirements Specification (SRS)

The SRS describes the requirements of the system and its external interfaces. Each requirement will be defined so that it is unambiguous and is capable of being objectively verified. The system requirements are based upon the user requirements, the Part 11 Assessment, and the System Risk Analysis.

#### Traceability Matrix

The Traceability Matrix identifies the relationships between each system requirement and its appropriate verification.

#### Architecture Document

This document describes the system architecture and high level design of the software. It identifies the architectural components or software items and the interfaces and communication mechanisms between the software items.

#### Detailed Design Description (DDD)

This document describes the low level design of the software and its decomposition into software units. The DDD includes functional and behavioral design descriptions of each software unit. These may consist of algorithms, state-event charges, class diagrams, or other graphical representations.

#### System Verification Plan

The System Verification Plan describes the test strategy for all system testing: the Installation Qualification (IQ), the Operational Qualification (OQ), and End to End Testing. IQ verifies proper upgrade of software on the servers located in the external hosting site. IQ also verifies proper installation of Agent software on the laptop. The IQ does not verify proper installation of the Client Application. The GBU responsible for the Client must perform an IQ on the Client installation. OQ verifies correct implementation of system and software requirements.. End to End Testing verifies the proper operation of the entire system, from medical device to server. The System Verification Plan shall describe the methods (e.g. inspection, analysis, simulation, demonstration, or test) and processes that will be used to perform system verification testing. It identifies the overall approach, and the features to be tested or excluded from testing.

#### System Verification Procedures

These documents contain the system verification protocols to implement the IQ, OQ, and ET. Each test procedure will list the tools needed to conduct the test and the pass/fail criteria. The System Verification Procedures may be scripted tests or exploratory tests, or a combination of both.

.

#### End to End Test Procedures

These documents contain the protocols used to verify proper operation of the entire system, from medical device to server. Each test procedure will list the tools needed to conduct the test and the pass/fail criteria.

#### Software Revision Level History

The Software Revision Level History lists each Release Candidate generated during development. For each Release Candidate, this document must include the person who created the Release Candidate, the date the Release Candidate was created, the version number and SHA hash code of each executable file, and a brief description of the changes from the previous Release Candidate. This document will be generated during the Development Phase at the time of the first Release Candidate and will be updated with each subsequent Release Candidate.

#### Source Code

The source code is generated based upon the design entities described in the Detailed Design Description. The code is used to implement the functionalities specified in the requirements specifications. The code will be generated during the Development Phase. It may be updated or new files created during the Test Phase.

#### Usability Evaluation Procedures

This document contains the protocols used to evaluate the user interaction and usability of the server GUI.

#### Phase Transition Report

The Phase Transition Report contains the results of the Phase transition reviews that are required at the end of each phase. It must list all documents created in the phase, their document numbers and current revision levels. It must list each review conducted during the phase and reference the review minutes which document the date of the review, outputs being reviewed, attendees, and any issues and their resolutions. It must identify the baseline generated at the end of the phase. It will be generated as a result of the Development Phase Transition review and shall be updated at each remaining phase transition.

#### Development Baseline

The Development Baseline identifies the approved deliverables that comprise the outputs of the Development Phase. It is documented in the Phase Transition Report.

### Test Phase Documentation

All Test Phase documentation will be generated during the Test Phase and approved prior to the Test Phase transition review. Documents will be updated as required throughout the software lifecycle.

#### System Verification Reports

These documents contain the conclusion reached as a result of each testing cycle. There are one or more System Verification Reports corresponding to each Release Candidate tested. The report shall include identification of the software being tested by Baseline Identifier, version numbers, and SHA hash codes, a description of any changes made to the software during testing including references to any issues and justification for any regression testing. It shall also include an exact description of the hardware configuration under test, and a concise statement of the conclusions reached as a result of the testing.

#### End to End Test Report

This document contains the conclusion reached as a result of the End to End Testing activities. The report shall include identification of the software being tested by Baseline Identifier, version numbers, and SHA hash codes, a description of any changes made to the software during testing including references to any issues and justification for any regression testing. It shall also include an exact description of the hardware configuration under test, and a concise statement of the conclusions reached as a result of the testing.

#### Regression Test Plan

This document describes the strategy for regression testing a new Release Candidate following a failed test cycle during System Verification testing. The strategy includes a list of which tests will be repeated and which will not be repeated, a list of which configurations will be tested, and which will not be tested, and a rationale for these decisions. The plan should document all defects that have been corrected in the new Release Candidate. If all defects found have not been corrected, a justification for this must be provided. The plan should document which existing test procedures will be re-executed, and identify any new test procedures created for the regression testing.

#### Regression Test Report

This document contains the conclusions reached as a result of executing the regression testing.

#### Software Verification Summary Report

This document contains the final summary conclusions reached as a result of executing all system testing.

#### Web UI Users Guide

This document is intended for Medtronic users of the Web UI of the GRSP server. It describes all user functions available through the Web UI. It does not cover the use of the Client Application.

#### Software Deployment Plan

This document describes the deployment strategy and plan for each release of the software. It identifies the versions of software to be deployed, deployment schedule and resources, any assumptions and risks relative to the deployment, and technical data migration and deployment methods. This document is created before the software is released.

#### Release Baseline

The Release Baseline identifies the approved deliverables that comprise the outputs of the Test Phase. It is documented in the Phase Transition Report.

#### OTS Software Validation Plan

This document describes the strategy and plan for the validation of the Off the Shelf (OTS) software used in the project. OTS software includes both software tools and SOUP.

#### OTS Software Validation Procedures

These documents identify the intended use of the OTS software, risk analysis, and test procedures. The existing legacy tools will be updatedaccording to [18] RL-010. The procedure is documented using [19] QD03461. All new OTS tools will be validated using NP042375 [22].

#### OTS Software Validation Reports

These documents contain the conclusions reached as a result of executing the OTS Software Validation Procedures. The report is documented using [19] QD03461.

#### Software Change Report

The Software Change Report is a list of all Jira issues that are part of a maintenance software release. For maintenance projects it describes the details of a modification to previously-released software.

#### Post Deployment Readiness Check Procedures

This document contains the protocols to be executed in the production system following deployment of a new version of software. All system verification protocols are executed in a QA environment and thoroughly exercise all aspects of the software. These protocols are to ensure that the software continues to work as intended after deployment into the production environment. It is a lightweight check of essential functionality.

#### Post Deployment Readiness Check Report

This document contains the conclusions reached as a result of executing the Post Deployment Readiness Check Procedures.

# SOFTWARE BUILD PROCESS

The Software Build Process is described in [6] Medtronic Gateway Build Process.

# SOFTWARE CONFIGURATION MANAGEMENT PLAN

Configuration management is a discipline to ensure the proper identification of the configuration, to control changes, and to record the change implementation status of the software.

## Configuration Management Tools

The Git version control system will be used to control the configuration of all software source code, development tools, and SOUP.

The Agile PLM system will be used to control the configuration of all project documentation requiring approval.

The IBM Rational Synergy version control system will be used to control the configuration of all MSMs.

The Configuration Management Tools shall store the history of the configuration items. Configuration items shall be retrievable.

## Configuration Items

All project deliverables listed in Section 5.2.

All development tools listed in Section 5.5.3

All SOUP listed in Section 5.5.4

## Responsibilities

The IUS Development team is responsible for configuration management activities associated with the source code, development tools, SOUP, and creation and labeling of baselines.

The IUS Project Manager is responsible for configuration management activities associated with project documentation.

## Change Control

As issues are discovered and changes are requested, the issue will be entered as an Issue into the Jira Issue Tracking system. All proposed changes will be reviewed by the Project Manager (PM) in consultation with the requester. If the proposed change has no impact on functionality, scope or schedule as determined by the PM, then the PM will accept or reject the change based on merit. If the proposed change impacts functionality, scope or schedule, then the change request will be forwarded by the PM for approval by the Change Control Board (CCB). This approval establishes permission for the Software Development team to make the change. The Issue Tracking system will be used to document the requested/approved changes and will constitute the Software Change Request.

### Change Control Board (CCB)

The CCB for the project will include the following roles: HI&M Product Owner, HI&M System Architect, SI Quality, Product Owner from each Client team.

### CCB Decision Criteria

There are three possible CCB decision for issues:

Approve

Reject

Defer

Criteria for Approve: Issue should be approved if the Priority is Blocker, Critical, or Major. Lower priority issues may be approved if there is time in the schedule to implement them.

Criteria for Reject: Issue should be rejected if it is a duplicate, cannot be reproduced, is no longer an issue, or is invalid. New feature requests may be rejected for a variety of business reasons as deemed appropriate by the CCB.

Criteria for Defer: Issue should be deferred if it is desirable to address the issue in a future release, or if it a defect that cannot be immediately duplicated but should be watched for re-occurrence.

## Software Versions

The Gateway Device Management Platform will have a Release numbering the format following [22]. When identifying a Version Number, two, three, or all four fields are referenced depending on the context in which the Version Number is used.

### New Device Integration Releases

Addition of new device will be considered as a minor release and will be odd numbered to distinguish it from regularly scheduled maintenance release. The details of versioning are outlined section 2.

### Planned Patch Releases

These are patches to the system to update underlying supporting systems such as Operating system update or security fixes. These will be done on an as needed basis, including during device release cycles, and will be verified through automated system tests.

### Emergency Patch Releases

In certain situations, the GDMP software may be updated in a patch release if a critical issue is found in the field which renders GDMP unusable or unstable. In such cases, additional testing besides automated testing may be required.

### Feature Releases

These are full scale MAJOR releases which add *major* new features (similar to Agile term “Epic”) to GDMP. These will be handled as new projects with associated development plan and schedule that will be determined when project is initiated.

### Release Tracking Plan

The system releases are labeled in the following format.

MAJOR RELEASE VERSION/**./** MINOR RELEASE VERSION/**. /**PATCH RELEASE VERSION/**.** /BUILD VERSION

The Major Release version for Epic type feature will increment by 1.

The Minor Release Version for Maintenance Release will be EVEN numbered.

The Minor Release Version for New Device Integration will be ODD numbered.

The planned Patch Release version will increment by 5 for each new patch release.

Emergency Patch Release will increment the currently released patch version by 1.

Build version will be updated as each new build is created during development and will be frozen when ready for testing.

**Example:**

For purposes of illustration, assume the currently released version on PROD is **4.1.5.xxx**

1. A new feature is released: the new version will be GDMP v **5**.0.0.xxx
2. A new maintenance version is released: the new version will be GDMP v 4.**2**.0.xxx
3. A new device integration version is released: the new version will be GDMP v 4.**3**.0.xxx
4. A planned patch is released: the new version will be GDMP v 4.1.**10**.xxx
5. An emergency patch is released: the new version will be GDMP v4.1.**6**.xxx

The reason for incrementing the Patch versions in increments of 5 is to allow adding unplanned security/system patches without impacting planned release numbering. For example, if a critical patch is required before planned patch release 4.1.5.x, it can be labeled 4.1.1.x, etc.

### Development Version: <MAJOR>.<MINOR>.<PATCH>.<BUILD>

Development Version is the term used for a release that is used exclusively by the development team. All four fields are specified when referring to Development versions. Development Versions are sometimes referred to as Engineering Versions. All 4 fields of the version number are maintained even though only 2 fields are typically referenced for use external to the company and 4 fields are referenced for internal company use.

## Configuration Identification

At the end of each phase in the software lifecycle, a baseline will be established that uniquely identifies the configuration items relevant to that phase. The baseline will be created in SVN and named according to the naming convention outlined in Section 10.6.1. Any deviation from the recommended baseline naming convention should be documented in the SDP. All configuration items comprising a release baseline shall be archived onto electronic storage media, and becomes part of the project’s Design History File.

Additionally, each configuration item will either be listed in or referenced from the Phase Transition Report. For source code, the Git baseline name shall be referenced from the Phase Transition Report, rather than listing all source code files.

### Mandatory Project Baselines

Each baseline will be given a unique SVN label that will allow retrieval of its configuration items. The table below shows mandatory project baselines and their naming conventions. Additional baselines may be created as desired by the project team.

|  |  |
| --- | --- |
| Project Milestone | Baseline Naming Convention |
| Transition from SW Development Phase to Test Phase | <project>-<date>-BL\_DEV\_X.Y.Z |
| Release | <project>-<date>-BL\_REL\_X.Y.Z |

### Optional Project Baselines

Additional baselines may be created as desired by the project team, for example, at the end of each sprint.

## Configuration Status Accounting

Configuration status accounting keeps a record of all the changes made to the previously released documentation or software baseline to reach the new baseline.

At the end of each development phase and at the end of a maintenance project, the Phase Transition Report shall list or reference the specific baselines applicable to that development phase or maintenance project, and the review minutes for any software reviews.

## Configuration Audit

Review and approval of the Phase Transition Report at each phase transition shall serve as verification of the baselines and the configuration management process.

# Branching Strategy

Start from GDMP4.0, one track branch strategy shall be implemented and used for current and future releases. The core idea is to simplify the GDMP source code repository maintenance effort. This one track branch strategy will make all the source code be developed and managed on the “Main” track with the CI (Continuous Integration) running against it to secure the quality and only create dedicated branch, call “release branch” for the required release when entering to formal testing stage. The latest changes are synced between the release branch and main track to make sure the main track always have the latest code with the best quality.

All the GDMP 4.0 code development is currently stored in the GIT on the main track. A CI machine is continuously running on it, deploy the latest code baseline and include the latest bug fix. When time is ready to decide to freeze the code baseline (“release candidate”) for the formal testing (including scheduled patch release, on boarding new device, emergency patch release), a release branch with the software version name convention described in 9.5 shall be created. A separate CI environment should be created to build and deploy the release candidate and any subsequent regressions. All subsequent regressions shall be tagged with the appropriate baseline according to section 9.5. Regressions shall be developed on the release candidate git branch. All the testing related to the release and its bug fixing shall be performed and validated on this release branch until it is released to Production.

# PROBLEM RESOLUTION PLAN

The Problem Resolution Plan defines the process for resolving problems detected in the software and software documentation. The Jira tool will be used to track issues for the project.

All defects, enhancement requests, requirements change requests, etc. shall be entered as Issues in the Jira Issue Tracking tool. All Issues will be investigated and resolved. When an Issue results in a change to the software, the change must be evaluated to ensure that the Issue is resolved, and additional problems are not introduced as a result of the change.

Issues, their resolution, and their verification shall be documented in the Issue Tracking tool, and this information shall be retrievable.

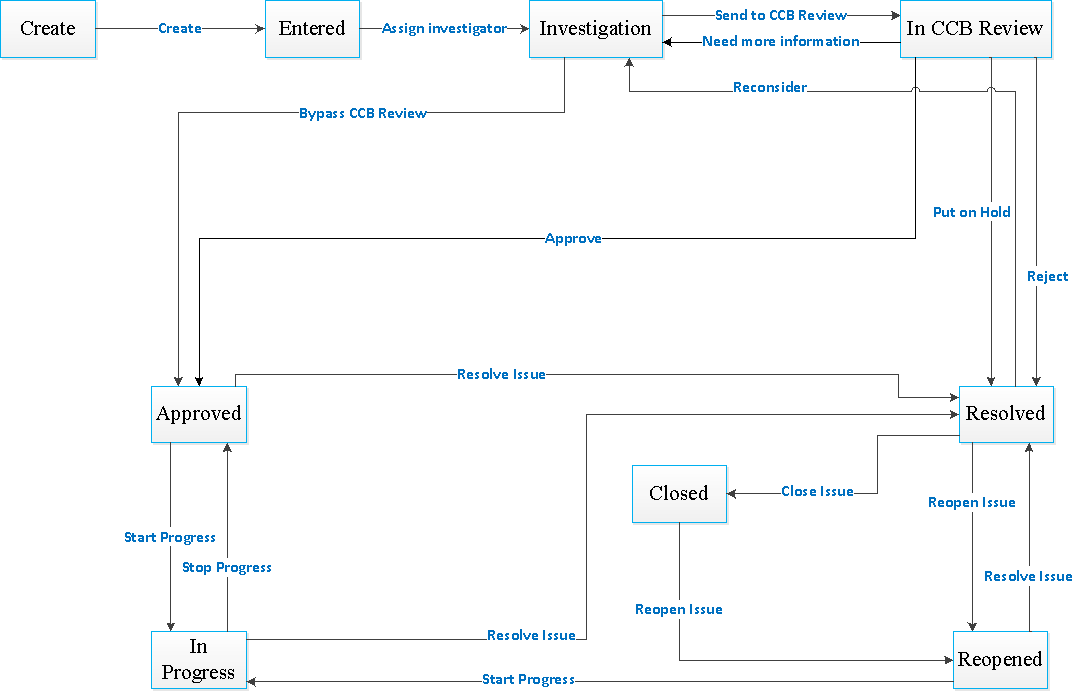
## Issue Workflow

The Issue Tracking tool implements a workflow through which each Issue is processed. There are two alternate forward paths through the workflow:

CCB approval needed

No CCB approval needed

|  |  |
| --- | --- |
| Path | When Used |
| CCB approval needed | For changes to approved requirements, except for minor changes that will be approved by the PM (e.g. clarification, re-wording, etc.)  For any change to software following transition into the Test Phase of the software lifecycle. PM approval can be substituted for CCB approval if the identified defects will be fixed by the development team prior to completion of testing and release of software. |
| No CCB approval needed | For any change to software during the Development Phase of the software lifecycle |



## Roles

|  |  |
| --- | --- |
| Role | Assigned Team Member |
| Submitter | Anyone |
| Project Manager | Leon Zhang |
| Investigator | Anyone on the Development Team |
| Resolver | Anyone on the Development Team |
| Tester | Anyone on the Test Team |
| CCB Member | See Section 9.4.1 |

## Issue States

|  |  |
| --- | --- |
| ISSUE STATES | |
| State | Description |
| Create Issue | This is the initial state of the Issue before it has been submitted. |
| Entered | The Issue has been submitted and is awaiting assignment for investigation. |
| Investigation | The Issue is being investigated. |
| In CCB Review | The Issue is under review by the CCB (or cross-functional team). |
| Approved | There are two reasons an Issue can be in this state:  The Issue has been approved by the CCB, but is not yet actively being worked on. (in Test Phase)  The Issue has been moved to Approved by the Investigator.(in Development Phase) |
| In Progress | The Issue is being worked on. |
| Resolved | There are three reasons that an Issue can be in this state:  The Issue has been resolved and is awaiting evaluation by the Tester.  The Issue has been rejected and is awaiting transition to Closed.  The Issue has been deferred (put on hold) and is awaiting transition back to Investigation to be reconsidered. |
| Reopened | The Issue has been reopened. |
| Closed | The Issue has been completed. This is the terminal/final state of the Issue. There are two reasons that an Issue can be in this state:  The Issue has been resolved and evaluated.  This Issue has been rejected and Tester agrees with rejection. |

## Issue Transitions

| ISSUE TRANSITIONS – FORWARD PATH | | |
| --- | --- | --- |
| Transition | Description | Who Transitions Issue |
| Create Issue  to Entered:  CREATE ISSUE | Submit the Issue.  An email notification is automatically sent to the person assigned to the Technical Lead role. | Anyone |
| Entered  Investigation:  ASSIGN | Assign Investigator from appropriate subsystem(s) to analyze the impact of the Issue. | Technical Lead |
| Investigation  In CCB Review: SEND TO CCB REVIEW | Investigation is complete and recommendations have been made to CCB (or cross-functional team, etc). | Investigator, in consultation with the PM |
| In CCB Review  Approved  APPROVE | CCB approves the Issue.  Assign a Resolver (could be the same as the Investigator) to work on the Issue and manage the Issue through the remainder of the workflow. | CCB Member (or PM) |
| In CCB Review  Resolved  PUT ON HOLD | CCB decides to defer the Issue. | CCB Member (or PM) |
| In CCB Review  Resolved  REJECT | CCB decides to reject the issue. | CCB Member (or PM) |
| Approved  In Progress:  START PROGRESS | Start working on the Issue. | Resolver |
| In Progress  Resolved:  RESOLVE ISSUE | Issue has been resolved and is ready for evaluation. | Resolver |
| Resolved  Closed:  CLOSE ISSUE | Issue has been evaluated and can be closed. | Tester |
| Resolved  Investigation:  RECONSIDER | Reconsider the original decision to defer the Issue by initiating a revised investigation. | Anyone |
| ALTERNATE FORWARD PATH | | |
| Investigation  Approved:  BYPASS CCB | Investigation is complete with one of three possible results: Approve, Put on Hold, or Reject. No CCB approval needed. This path is appropriate only during the Development Phase. | Investigator |
| Approved  Resolved  RESOLVE ISSUE | Issue is either deferred or rejected. No CCB approval needed. | Investigator |
| BACKWARD PATHS | | |
| In CCB Review Investigation  NEED MORE INFORMATION | CCB needs more information to make a decision. | CCB Member (or Project Leader) |
| In Progress  Approved:  STOP PROGRESS | Suspend work on Issue. | Resolver |
| Resolved  Reopened:  REOPEN ISSUE | Issue was not resolved and is being reopened | Tester |
| Closed  Reopened:  REOPEN ISSUE | Issue should not have been closed and is being reopened | Anyone |
| Reopened  Resolved:  RESOLVE ISSUE | Issue is resolved without any additional work needed and is ready for evaluation. | Resolver |
| Reopened  In Progress:  START PROGRESS | Start work on reopened Issue. | Resolver |

## Issue Priority Levels

|  |  |
| --- | --- |
| Priority | Description |
| Blocker | Blocks development and/or testing work, production could not run. |
| Critical | Crashes, loss of data, severe memory leak |
| Major | Major loss of function. |
| Minor | Minor loss of function, or other problem where easy workaround is present. |
| Trivial | Cosmetic problem like misspelt words or misaligned text. |

## Issue Types

One of the following types will be assigned to each issue in Jira.

|  |  |
| --- | --- |
| Type | Description |
| Defect | Feature does not behave as specified |
| Improvement | Request for a change in the behavior in an existing feature |
| New Feature | Request for a new feature, not already covered by any existing requirements |
| Change Request | Request a change to an approved requirement |

## Software Tool Anomalies

Anomalies found in the software tools shall be documented in an anomalies.txt file stored with the tool in Git.

## Trend Analysis

After product release, customer complaints will be analyzed to identify trends or patterns of software problems. This data will inform improvements in both the software and the software development process.

# QUALITY PLAN

The following verification and validation activities will be performed:

* Code reviews of all custom source code (excludes SOUP). When necessary, code review issues will be fixed during the review. Otherwise, they will be entered as an issue in Jira to be prioritized and fixed in a subsequent sprint or release. Code reviews will not be documented in a report.

NOTE: During the code review, the reviewers should be aware of whether the changes will result in modifications to the deployment of the system beyond the simple updating of the application binaries. The JIRA ticket which is attached to the changes being reviewed must be labeled with “DEPLOYMENT\_IMPACT.” A sub task must be created to modify the deployment system (SOUP repository, build dependencies, SQL scripts, instructions, cookbooks, recipes, etc).

* Unit testing with a goal of at least 70% coverage of custom code (excludes SOUP). Unit testing will not be documented in a report.
* System verification testing (IQ, OQ, ET)
* End to End testing
* Validation of all SOUP used in the system
* Validation of all tools used to develop and test the software

## Test Strategy

Scripted testing (pre-approved test procedures, either manual or automated) will be used for the IQ, OQ, End-to-End Testing, Database Validation, and verification of all requirements that mitigate risks identified in the Risk Analysis (critical requirements). ET may be used for additional coverage of functionality tested with scripted test procedures. Existing scripted test procedures used during previous testing of the GDMP will be re-used even if they cover non-critical requirements.

Automated testing will be utilized for all the tests that can be automated. The tests, test subsections that cannot be automated will be executed manually by the tester.

The individual GBUs are responsible for validating their user requirements, and evaluating usability of their Client Applications.

# RELEASE CRITERIA

## Criteria for Release to System Verification Testing

* All features are implemented.
* Code reviews are completed.
* All Blocker, Critical, and Major defects in Jira found during Development phase are resolved.

## Criteria for Release to Production

* All deliverables identified in this document are released.
* All Blocker, Critical, and Major defects in Jira are resolved.