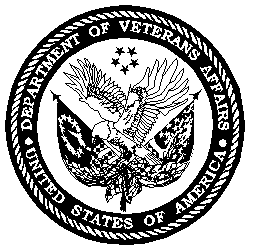
<Enter Project name on the next line>

Configuration Management Plan



Version 1.0

Date

Department of Veterans Affairs

Revision History

This template is designed for duplex printing.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Introduction 1](#_Toc261508357)

[1.1. Foreword 1](#_Toc261508358)

[1.2. Scope 1](#_Toc261508359)

[1.3. Audience 1](#_Toc261508360)

[1.4. Overall Description 1](#_Toc261508361)

[1.5. CM Mission Statement 2](#_Toc261508362)

[1.6. CM Mission Objectives 2](#_Toc261508363)

[1.7. Reference Documents 3](#_Toc261508364)

[2. CM Organization 4](#_Toc261508365)

[2.1. Configuration Management Team 4](#_Toc261508366)

[2.2. Change Control 6](#_Toc261508367)

[2.2.1. Different Levels of CCBs 7](#_Toc261508368)

[3. CM Activities 9](#_Toc261508369)

[3.1. Configuration Identification 9](#_Toc261508370)

[3.1.1. Documentation Identification 10](#_Toc261508371)

[3.1.2. Documentation Categorization 11](#_Toc261508372)

[3.1.3. Software Identification 11](#_Toc261508373)

[3.1.4. Change Identification 11](#_Toc261508374)

[3.1.5. Release Identification 11](#_Toc261508375)

[3.1.6. Release Candidate Identification 12](#_Toc261508376)

[3.1.7. Build Identification 12](#_Toc261508377)

[3.1.8. Baseline Identification 13](#_Toc261508378)

[3.2. Configuration Control 14](#_Toc261508379)

[3.2.1. Interface Management 14](#_Toc261508380)

[3.2.2. V*IST*A Package Release 15](#_Toc261508381)

[3.2.3. Document Management Strategy for Project Documents 16](#_Toc261508382)

[3.2.4. Change Management 16](#_Toc261508383)

[3.2.5. CCB Review Process Flow 18](#_Toc261508384)

[3.2.6. The Complete Change Control Process 18](#_Toc261508385)

[3.2.7. Issue Management 18](#_Toc261508386)

[3.2.8. Change Management Request 19](#_Toc261508387)

[3.2.9. Change Request Form 20](#_Toc261508388)

[3.3. Status Accounting 20](#_Toc261508389)

[3.3.1. Status Reporting on Change Request Activity 21](#_Toc261508390)

[3.3.2. General Status Reports 21](#_Toc261508391)

[3.4. Configuration Audits and Reviews 21](#_Toc261508392)

[3.4.1. Build Audits 21](#_Toc261508393)

[3.4.2. Testing Audits 21](#_Toc261508394)

[3.4.3. Test Readiness Review 22](#_Toc261508395)

[3.5. Release Management 22](#_Toc261508396)

[3.5.1. Release Packaging Concepts 22](#_Toc261508397)

[3.5.2. Electronic Release 23](#_Toc261508398)

[3.5.3. Version Description Document 23](#_Toc261508399)

[3.6. The System Development Life Cycle 23](#_Toc261508400)

[4. CM Environment 25](#_Toc261508401)

[4.1. Development Tools 25](#_Toc261508402)

[4.2. CM Tools 25](#_Toc261508403)

[4.3. <PROJECT> CM Environment Roadmap 26](#_Toc261508404)

[4.4. Requirements/Initial Conditions 26](#_Toc261508405)

[4.5. CM Architecture for the <PROJECT> Development Environment 26](#_Toc261508406)

[4.6. CM Methodology – Distributed Development 26](#_Toc261508407)

[4.7. Repositories 26](#_Toc261508408)

[4.7.1. Status Accounting Repository 26](#_Toc261508409)

[4.7.2. Development Environment Repository 27](#_Toc261508410)

[4.7.3. Project Management Repository 27](#_Toc261508411)

[4.8. Build Tools 27](#_Toc261508412)

[4.8.1. Component Builds 27](#_Toc261508413)

[4.8.2. IDE Integration 27](#_Toc261508414)

[4.8.3. IDE Configuration 27](#_Toc261508415)

[4.9. Guidelines for data organization 28](#_Toc261508416)

[4.9.1. Component Hierarchy 28](#_Toc261508417)

[4.9.2. Access Control 28](#_Toc261508418)

[4.9.3. Access List 28](#_Toc261508419)

[4.9.4. Typical User Actions Within the CM Environment 29](#_Toc261508420)

[4.9.5. Development Workspace (Work Area) 29](#_Toc261508421)

[4.9.6. Development Tool Updates 29](#_Toc261508422)

[4.9.7. Migration through the SDLC for the <PROJECT> Project 30](#_Toc261508423)

[4.9.8. Managing the Turnover Process 31](#_Toc261508424)

[4.9.9. Parallel Development 31](#_Toc261508425)

[4.9.10. Branching 31](#_Toc261508426)

[4.9.11. Merging 31](#_Toc261508427)

[5. CM Processes and Procedures 32](#_Toc261508428)

[5.1. The CM Process 32](#_Toc261508429)

[5.1.1. The CM Compliant Process Defined 32](#_Toc261508430)

[5.1.2. CM Process Flow Model 33](#_Toc261508431)

[5.1.3. Backup and Disaster Recovery Plans 34](#_Toc261508432)

[5.1.4. Method for CM Environment Configuration Recovery 34](#_Toc261508433)

[5.1.5. Method for <PROJECT> Site Servers Recovery 35](#_Toc261508434)

[5.1.6. Weekly HRPC Backup and Recovery 35](#_Toc261508435)

[5.1.7. Communication and Environment Coordination 35](#_Toc261508436)

[6. Acronyms, Abbreviations, and Terms 36](#_Toc261508437)

[6.1. Acronyms and Abbreviations 36](#_Toc261508438)

[6.2. Terms 37](#_Toc261508439)

[A. Attachment A Approval Signatures 40](#_Toc261508440)

# Introduction

## Foreword

This document is intended to provide a uniform approach to Configuration Management (CM) for <PROJECT>. It describes the processes and procedures for managing and controlling the development, delivery, and maintenance for <PROJECT> applications throughout the project’s application Development Life Cycle (SDLC). This plan also outlines the CM organization, responsibilities, activities, configuration change control flow, as well as the CM requirements to be followed by all project teams. This CM Plan will aid both management and development staff in the production of a high-quality software application.

HSD&D is currently using HealtheVet VistA Development and **V**IST**A** Development (Legacy Systems). This CM Plan will address both HealtheVet **V**IST**A** Development and **V**IST**A** Development. The CM Plan is being developed under two phases. The first phase focuses on the HealtheVet **V**IST**A** Development. Phase two will incorporate **V**IST**A** items.

## Scope

The CM environment definitions outlined in this document apply to all activity conducted within and for the <PROJECT> project, as related to the <PROJECT> product. The CM processes and procedures outlined in this document apply to all software (including COTS software), and documentation including tracking, auditing, verification, and version description documentation. This CM Plan will cover the performance and management applicable to CM activities including configuration identification, interface management, configuration control, status accounting, audit and verification, releases, and baselines for the project.

The information contained in this document is most applicable to the development of the application. However, it is also intended to serve as the model by which it will be delivered, maintained and enhanced in the future. Any processes and procedures defined and documented for use within the CM Environment described in this document must meet all the criteria of a formally documented process for inclusion and support by the CM environment and its associated backup and recovery utilities.

The information contained in this document is most applicable to the day-to-day performance of activities within the CM environment; however, it can and should be used as a basis for implementing ongoing business process improvement as needed throughout the lifetime of the project. This document is NOT intended to govern change directly pertaining to the Operations and/or Deployment teams.

## Audience

The primary audience for this document consists of all <PROJECT> project team members or others, who have a requirement, need, or desire to develop, change, fix, or enhance the <PROJECT> applications or project artifacts.

## Overall Description

The implementation of a formal and structured CM Environment ensures that all <PROJECT> project artifacts are baselined and maintained in a stable environment. It provides the ability to apply state-of-the-art technology and environment enhancements to satisfy all projects plan development and test objectives, which maintains the integrity of the project deliverables.

The complexity involved in the configuration management process requires the implementation of formal processes and procedures. These processes and procedures provide objectives, requirements, and step-by-step instruction on the performance of activities within the CM Environment for this project. This document will provide discussion on topics such as the organization of personnel and the division of tasks for the project team. The processes and procedures are documented in the Change Management Processes and Procedures document, and referenced in this CM plan.

The term change request (CR) in this document is a generic term used to indicate, or refer to, both defects and enhancements.

## CM Mission Statement

Design, implement, maintain, and support the project CM policies, processes, and procedures in accordance with the SDLC, and with the following primary purposes in mind:

* Maintain accountability for all activities that impact “baselined” environments and data for the project.
* Maintain and Protect the Veterans Affairs substantial software investment.
* Provide a secure means of support for concurrent/parallel development and test.

## CM Mission Objectives

The following objectives meet the established policies and procedures to meet the mission statement, and to benefit future investments in new product development. This increases the benefit per investment dollar margin, and realizes a more efficient SDLC, while maintaining stable and reliable deliverables for the <PROJECT> team.

* Establish and maintain integrity of the baselined artifacts for the project.
* Implement and maintain version management on all aspects of hardware, software, and documentation that leads to the release and implementation of a released configuration.
* Enable the delivery of any revision level for any test records, module or product.
* Enable product configuration status reporting and accounting.
* Establish and maintain repeatability of development and quality assurance (QA) processes using the appropriate audit processes.
* Establish and maintain a minimum number of common repositories.
* Provide and maintain tools for problem tracking status relevant to each product configuration.
* Automate software build processes without compromising the integrity of the build environment while maintaining or improving the quality of the build.
* Automate software development processes without compromising the integrity of the development environment while maintaining or improving the quality of the developed products.
* Provide avenues of training for users of Configuration Management tools and version control products.
* Establish and maintain communications with the user community regarding the CM activities. Methods may include scheduled forums, newsletters, Web page, etc.
* Establish and maintain Configuration Management standardized processes for the SDLC.

The objectives specific to this CM Plan and the project include the following:

* To assist management in achieving, at lowest total life cycle cost, the required performance, operational efficiency, logistics support, and readiness of configuration items.
* To allow the maximum degree of design and development latitude and introduce, at the appropriate time, the degree and depth of configuration control necessary for production and maintenance support.
* To attain the maximum efficiency in the management of changes with respect to their necessity, cost, timing, and implementation.
* To attain the optimum degree of uniformity in configuration management policy, procedures, data, forms, and reports at all interfaces.
* To assure that the functional and physical characteristics of configuration items, as established by the technical documentation, for the product are recorded. (via a Configuration Management audit)

Configuration Management functions specific to this project to be performed in addition to the stated objectives are:

* To manage changes to project artifacts including models, software, and documentation
* To collect metrics
* To generate progress reports
* To build executables for software testing and deliverables
* To provide status accounting reports

## Reference Documents

Reference documents are those documents that, although not a part of this document, serve to strengthen and clarify its contents. The following documents provide reference material for background information only. In case of conflict, the senior document for project requirements and protocol is the <PROJECT> Iteration Plan; otherwise, the Configuration Management Plan will take precedence. Contact the CM team for instructions on how to obtain copies of the released version(s) of these documents.

* <PROJECT> Software Development Plan
* <PROJECT> Requirements Management Plan
* <PROJECT> Analysis Artifact Standards Document
* <PROJECT> Design Standards Document
* <PROJECT> Software Architecture Document
* <PROJECT> Vision Document
* <PROJECT> Build Guide
* <PROJECT> Master Test Strategy
* <PROJECT> Deployment Plan
* <PROJECT> CCB Charter
* <PROJECT> Implementation Plan
* Veterans Health Administration, Tier 1 Regional Processing Center, Performance Service Level Agreement (PSLA)”, Section 1.01, for Rational Enterprise Suite

# CM Organization

This section defines the CM tasks performed by all teams. The following team members participate in the delivery of artifacts (code, documents, etc.)

* Project Management
* Development Team Leads
* Change Control Board (CCB)
* Developers, Functional Analysts/Technical Analysts
* Technical Writers
* SQA Team (QA Process Manager, SQA Analyst, SQA Test Analyst)
* Deployment Team -HSITES
* Implementation Team
* Data Architecture Representative/DBA
* EVS
* Operations Team

## Configuration Management Team

The <PROJECT> project is managed and organized, as an integrated project team comprised of development personnel, located in the Veterans Health Administration’s (VHA) facilities and other offsite development locations, such as approved contractor/vendor, Southwest Research Institute (SwRI).

Table 1 lists the personnel who are members of the project team and CM team along with their assigned roles and responsibilities. The Roles defined herein can sometimes be overlapped with other roles and responsibilities depending on the environment. In addition, one person allocated for a specific role as listed below may often have the responsibility of other roles.

Table 1: Configuration Management Team Roles and Responsibilities

| Role | Responsibility |
| --- | --- |
| Project Management/ Change Control Manager | Ensures proper execution of the CM Plan.  Oversees the configuration management process.  Approves the internal change requests.  Assesses and evaluates all other change requests.  Attends Change Control Board (CCB) meetings. |
| Configuration Manager/CM Team | Educates project team members in CM “best practices.”  Develops and maintains CM plan and procedures.  Establishes, promotes, and releases baselines.  Creates and administers CM repository.  Builds and manages CM tool environment.  Performs interim and final builds.  Prepares release package, release archives and version description documents (VDD).  Accountable for instituting the established processes and reporting progress statistics based on change requests.  Attends CCB meetings.  Identifies project baselines as necessary.  Responsible for CM audits and necessary status accounting related to the project.  Conducts audits of scheduled milestones. |
| Development Team Lead | Submits build/release requests.  Coordinates development activities and assigns tasks.  Ensures all CM procedures are implemented and followed for all software, documentation, and/or any other components for which they are responsible.  Ensures all developers’ work within the specified CM process and related guidelines as specified in the Change Management Processes and Procedures document.  Attends the CCB meetings and provide technical details, as required. |
| Developers/Functional Analysts/Technical Analysts | Follows the CM Plan and Procedures: including, but not limited to:  Unified Change Management (UCM) version control procedures.  Maintain accurate, detailed information for all assigned change requests (CRs), in the CR database, related to the applicable development detail of the CRs lifecycle.  Provide impact analysis reporting for the CCB approved problems or changes, including documentation of suggested solutions to facilitate CCB disposition activities.  Documentation of build, release, and installation Instructions. |
| CCB | Governing body for reviewing and approving change requests under the CM process. (See section 2.2 for details) |
| Technical Writer | Develops technical deliverable documentation to support the software. Provides editing, formatting, and graphics support for documentation. This team consists of the technical writers. |
| SQA Manager | Ensures all SQA Analysts work within the CM processes and procedures.  Verifies that only CM-approved releases are installed into the test environment(s).  Ensures that SQA Analysts are always testing from official CM releases.  Attends CCB meetings and provides testing details, as required.  Reviews status accounting related to the project.  Reviews released artifacts. |
| SQA Analysts | Responsible for testing installed releases, as CM provides releases from development.  Update CRs assigned to them according to test activity results.  Determines Pass/Fail for each CR scheduled for a release.  Opens defect CRs for any newly discovered problems during testing. |
| Deployment/HSITES <need to verify this> | Coordinates the release and deployment of software to the existing sites and the newly activated sites. |
| Implementation Team | TBD |
| Database Architecture Rep/DBA | Manages the database design and development of the databases. |
| System Administration (Infrastructure) | Manages the development and test network environment and works with the development teams to ensure correct network configuration. |
| EVS | TBD |
| Operations | TBD |

## Change Control

Change Control is typically defined as the ability to coordinate, track, and manage change to the baseline of an application’s (or project’s) configurable items. Change Control for a project within the VAOI should satisfy all Change Management criteria established by and for each project. These criteria should include the capability to satisfy all project plan development and test objectives while maintaining the stability and integrity of the product baseline within the project.

The complexity involved in the Change Control process requires the implementation of formal processes and procedures to be documented in detail for each project in the Change Management Processes and Procedures document. These processes and procedures provide objectives, requirements, and step-by-step instruction on the performance of Change Control activities within the CM Environment for each project. The Change Management Processes and Procedures document will also provide sections with information on key process areas such as:

* The organization of personnel and the division of tasks for the project team.
* The commissioning of various levels of review such as a Technical or Project level Change Control Board (CCB).
* Decision criteria and escalation requirements specific to each level of change request review.
* The overall Change Management Process description and flow.

For each CCB, the typical project Change Control Board structure includes:

* Scope (level of authority to approve change)
* Objectives
* Escalation
* Roles & responsibilities
* Representatives
* Entry/exit criteria

### Different Levels of CCBs

#### Technical CCB (cross-project service line CCB) (TCCB)

Review/Approve/Disapprove/Escalate software change requests (enhancements, defects), new service requests (NSRs), and patient safety issues.

Escalate changes not affecting schedule, budget, and current resource allotments, interfaces external to which this change is commissioned.

Subject to implementation of changes as directed by HSD &D CCB.

#### HSD & D CCB (Internal)

Review/Approve/Disapprove/Escalate software change requests (enhancements, defects), NSRs, and patient safety issues that have been escalated from the Technical CCB.

May approve changes affecting schedule, budget (to a specified dollar amount), and current resource allotments.

Escalate changes related to interfaces external to which this change is commissioned.

Subject to implementation of changes as directed by HSD &D CCB.

#### CM Review Board

The CM Review Board is commissioned for the primary purpose of managing configuration and change requests to the CM environment itself and assessing impact on the artifacts produced within that environment.

The CM Review Board is staffed from the CM Team and on occasion, other guest project personnel. The CM Review Board brings together the key decision makers for the project, which are involved in day-to-day development and gives the Project Manager an opportunity to review and/or present any changes that may impact the schedule, cost or quality of the project.

#### Environment Change Control Board (ECCB)

This board analyzes changes to the framework, tool suite, etc., and makes recommendations to the HSD &D Change Control Board. In addition the ECCB can:

Control the development environment.

Review white paper of recommended tool changes.

Review/approve changes to HSD &D development standards, which includes:

GUI development standards

Database enterprise model and data element naming conventions

Use of standard object library components

Framework code generation standards

Software Development Notebook standards

Development Tool

# CM Activities

The primary goal of Configuration Management’s Environment Structure and processes and procedures, entails configuration identification, interface management, configuration control, status accounting, project auditing, product verification, and release management.

Figure 1 illustrates how the process flow parallels the Iterative Development Lifecycle (IDL) Software Development Process (SDLC) of the <PROJECT> development methodology.

Figure 1: CM Process in Relationship to the Software Development Methodology

|  |
| --- |
|  |

## Configuration Identification

Configuration identification is the process of identifying, selecting, naming, and classifying the development items subject to change control. This process consists of identifying, selecting, and documenting the following items and any other items pertaining to the following that are to be placed under configuration management control:

* Point in the development in which the application (or subsystem) will be placed under configuration management
* The Project itself - unique identifier assigned to the application
* Subsystems - characteristics of the subsystems
* Baseline to which the subsystems belong

These items consist of the set of currently approved or conditionally approved technical documentation, modeling artifacts, source code, executable image, and object files that identify and describe the functional and physical characteristics of the application.

The CM team will work with the Project Manager to select the appropriate identified configuration items (CI), then document and maintain these items under configuration control, including and/or related to the following:

* Documentation/Models
* Change Requests
* Application software versions
* Repositories
* VistA patches
* COTS product versions
* Test configurations and test documentation
* Builds
* Baselines
* Releases

Configuration Identification information will be recorded in each released Version Description Document (VDD). Please note the following identification items are for a typical CM process.

### Documentation Identification

All project related documents will be managed and controlled in the designated documents repository for that type of document. The following list identifies some of the types of documentation to be managed:

* Project administrative artifacts
* Software development processes and procedures
* COTS product versions
* Requirements, design, and architecture documents
* Software test plan, procedures and automation scripts
* Configuration management documents (Configuration Management Plan, Change ManagementProcesses and Procedures, and CM Environment documentation)
* Deployment plan (The plan for deployment implementation)
* Installation Guide (Processes and procedures for installing the delivered application package)
* User Release notes (As applicable, to provide detailed information about new features, enhancements, changes, bug-fixes etc included in a particular release)
* Technical Release notes (Technical information provided by the development team to assist in updating the application with the new release package. It works in conjunction with the deployment process and procedures. For example: database scripts that need to be run, new executables that need to be loaded, and changes to configuration files. It also contains the status of Change Requests pertaining to the release)
* Version Description Document (VDD)

### Documentation Categorization

Any documentation artifact should be categorized as either Project Management or Development. The criteria are relatively simple should make it easy to determine which category and ultimately repository a new artifact should be placed in.

#### Project Management Artifacts

Project Management artifacts are typically those that do not require migration through a series of life cycle stages, but rather, once generated are static and relate to a specific state that the project is in at a certain time within its life cycle. For example, Software Development Plan, Status Reports.

#### Development Artifacts

Development Artifacts are those that change along with the developed product and most likely will experience change over time and thus require baselining and Change Management. They also typically represent specific configurations and criteria required by the developed release of the product to which they are associated with and thus should be packaged and migrated through the same series of life cycle stages as all other baselined artifacts within the development environment architecture and utilities.

### Software Identification

Software identification is the version or revision identifier of each software item, which makes up the whole of the <PROJECT> application at any specific baseline. It allows identification of unique items that make up the inventory of a baseline. Software identification provides the foundation for traceability of related artifacts. The software identification is assigned by the version control automated tool and controlled in the development repository. Each VDD lists all software, its version, and related change document numbers.

### Change Identification

Change Identification is acquired through change documentation identifiers, which are assigned automatically by the CM tool and are controlled and tracked in the status accounting repository. Change Request methodology and lifecycle states are outlined in the subsequent sections of this document.

### Release Identification

A <PROJECT> Release is the distribution of the baseline or application modifications to the baseline. A Release is comprised of one or more software or documentation modifications authorized for the implementation/correction issued by change requests. Distribution of a release is delivered with a VDD. A release is a way of identifying the exact version of all objects that comprise a product at a certain point in time.

Releases are identified by the ICCB and approved by the CCB. A release is identified by a combination of a release number and version number. The naming convention for a Release depends on the type of release being delivered. There are three types of releases:

Major - Releases are designated as major if the application functionality is significantly altered.

Minor - A release is defined as minor for all other fixes, enhancements or maintenance and provides backward compatibility to previous releases.

Patches - Patch releases (or Release Updates) are delivered only in emergency cases for high priority problems that may impact the application’s production environment execution or processing capabilities.

The <PROJECT> CM Team will define the identification and composition of a specific release. The Production Release identification (ID) nomenclature will meet the following criteria:

Each release id will consists of a 3 part positional parameter naming convention consisting of the following components. Ex: “R1I1U1”, “R2I1U3”

The first component is the Major release and is identified by a capital letter R (for Release) followed by numerically incremented digit(s) representing the next consecutive numbered release.

The second component is the Minor release is identified by a capital letter I (for Iteration) followed by numerically incremented digit(s) representing the consecutive number of iterations that have been applied since the last delivered major release. A newly packaged Major release will always be designated as the first iteration (I1) of that Major release.

And the patch release (or Release Update) identified by a capital letter U (for Update) followed by numerically incremented digit(s) representing the consecutive number of updates that have been applied to a particular baselined release.

Utilizing the release id nomenclature described above, a possible id for a newly packaged release would be: “R2I3U5” which indicates that the third iteration of release number two with update number five applied is what makes up this particular package. A Production Release may consist of one or more Release Candidates.

### Release Candidate Identification

When used, the Release Candidate identification (id) nomenclature should meet the following criteria:

Each candidate that could be promoted to a Production Release has an ID identified by 2 capital letters RC (for Release Candidate) followed by numerically incremented digit(s) representing the next consecutive numbered release candidate. Ex: “RC\_1”, “RC\_2”, etc.

Delivery into the SQA test phase will change a build into a Release Candidate. The Release Candidates may or may not be implemented as an individual Production Release or any number of them can be assembled into a Production Release.

### Build Identification

Each build label will represent a date and time sensitive baseline ID within the SCM environment to support logging, reporting, and restoration capabilities. Based on the success of a build, deployment identification of a build for testing will be consistent with the release naming convention described in the previous section and assigned appropriately.

Builds will be performed in and for several environments. Builds originating from the development group, also being built by the CM team, will conform to these build instructions. Builds necessary for the development process will be the responsibility of the development team.

There will be 2 types of builds; package builds (Components) which are builds of the individual application components and system builds (Release Packages) which would include all the packages, constituting the entire release package for a specific environment. To assist in tracking a Build the following naming convention has been established.

Package Build:

|  |  |
| --- | --- |
|  | <Component>\_BLD\_<#> |
|  | Component = Package being built. |
|  | BLD = it is a build |

# = (sequential order) unique identifier of which time the build is being performed. For example: Framework\_BLD\_03

System Build:

|  |  |
| --- | --- |
|  | <Environment>\_BLD\_<#> |
|  | Environment = designated location that the build of the application is intended |
|  | BLD = it is a build |

# = (sequential order) unique identifier of which time the build is being performed. For example: Dev\_BLD\_5

This becomes the release ID of the product/application deployed within a given environment until the builds become a Release Candidate.

**Naming for Projects with a Single Build type:**

**Formal Build**:

|  |  |
| --- | --- |
|  | BLD\_<date and time in the format: ddmmyyyyhhmmss> |
|  | BLD = it is a build |
|  |  |

At any time after a formal build has been completed it may be incorporated into a Formal Release Baseline.

### Baseline Identification

Since a baseline is an aggregate of a group of configuration items, it can itself be classified as a configuration item. Assignment of a unique identifier to a baseline will be accomplished in one of two ways:

By the creation of a unique label based on date and time created and applied. Each build or release ID will always be associated with the baseline IDs it consists of.  
  
For Example: <PROJECT> or Sub-Project Name\_Rel\_1.0\_Itr4\_Build4

By utilization of the Release ID as discussed in section 3.1.5 which will then be considered as permanent on the object versions to which it has been applied. Thus the combination of the formal build id, the Release ID and status labels such as those for “approval” and stages “DEV”, “TEST”, and “PROD” will provide information needed for tracking all releases through the SDLC.

## Configuration Control

Any product development process (especially object-oriented software) is an iterative, incremental one. Software, documents, environment structures, hardware configuration, operating system, and system tools must all be managed to provide a stable and dependable environment for the creation and implementation of the intended products. Configurable items in the scope of this document are defined as any object (software, hardware, documentation, environment configurations, etc.) that may experience change over time.

The act of creating any software deliverable implies that objects will be created and modified in the design and development process, which by definition become project CIs along with the environments and documentation developed to support them. After development is complete, testing for validation and certification is required. This can only be accomplished accurately if the integrity of the software deliverable can be maintained through formal change control.

Formal change control involves the management of changes and versions to the CIs. Additional concerns associated with change control are of product migration beyond development, to the various stages of testing and certification, and until it is released into a production environment.

This is commonly referred to as the Product Development Lifecycle, and if it primarily deals with a software product, it is sometimes termed the System Development Lifecycle or SDLC. All SDLC activities performed to satisfy any of the configuration control functions are supported within the formal CM Environment as described in this section.

### Interface Management

Interface management is captured at the Enterprise level and at the Project level.

#### Enterprise

Determine the group or individual responsible for tracking and communicating the enterprise interface infrastructure. For legacy ***V****IST****A*** projects, this is managed through the [VistA Maintenance Project](http://vhaispvmp9.vha.med.va.gov/). See the website: <http://vhaispvmp9.vha.med.va.gov/> for additional information. The objectives specific to interface management include:

Identify all internal and external interfaces for One-VA.

Document all interface standards.

Develop One-VA common interface model representation for HW/SW.

Publish interface structures and model representation in a standard enterprise document that is centrally controlled.

Keep documentation up to date based on feedback from projects.

Develop enterprise level single source configuration/change tracking system to manage enterprise interfaces and the projects these interfaces affect.

Develop audit process to ensure conformance to interface standards.

#### Project

Using the published interface structures and model representation documentation, identify interfaces affecting product deliverables for the project. Document interface dependencies, and develop a model describing all applicable project interfaces, such as:

Internal (**V***IST***A**, MVR, MPI etc.)

Determine patch dependencies for namespaces

External

Future

Notify the project TCCB and enterprise interface management representative of all interface dependencies and their potential impacts. If project interfaces are identified that have not been accounted for in the enterprise’s published interface structures and model representation documentation, notify the enterprise interface management representative.

Determine responsible parties for each identified interface.

Determine mechanism for communicating interface activities (e.g., ICWG).

Determine Tracking and Control Mechanism, such as:

Project Dependency spreadsheets

Interface Agreements

Forum

KIDS builds

The CM team is responsible for tracking and controlling the software and interface development process amongst multiple team members and often across multiple projects. The Tracking and Control Mechanism must support geographically separated team members and multiple development environments.

### V*IST*A Package Release

**V***IST***A** interface components are developed and maintained according to the Kernel Installation and Distribution System (KIDS) and the National Patch Module (NPM) Operational Summary. The **V***IST***A** Package Release Process includes the following activities and checklists:

Package Creation Preparation

Package Release Preparation

SQA Patch Checklist approved by Project Manager

Package/Patch Completion Transition Document

KIDS Preparation Checklist

KIDS Build Checklist

Packages and patches are tracked on the VA network using Patch Cover Sheets. All **V***IST***A** packages must undergo 5 Enterprise **V***IST***A** Services (EVS) Review Points including:

Capacity Planning

Technical Review

Data Dictionary

Health Level 7/Messaging

Integration Agreement

### Document Management Strategy for Project Documents

The design for Document Management that uses a Project Management Repository and or other tool specific repository will be used for the <PROJECT> project. The only artifacts that should then be stored within these Project Management Repositories are those directly related to the Project Management effort such as:

Project Schedules

Meeting Agendas

Meeting Minutes

Project Plans

Status Reports

Project Document Templates

### Change Management

Change Management for software projects must combine necessary control with rapid and flexible response to change. At the earliest stage of development, the project development team is responsible for change management until artifacts are submitted to the CM staff for formal change control.

Formal change control begins when the Project Manager and/or Team Leader determines a significant risk to the application’s integrity exists if adverse and or unrecoverable changes are introduced.

#### Change Management for <PROJECT>

A key principle in the change control process is to push the change authority to the lowest possible level while still maintaining sufficient control. Change authority depends upon the phase of development the project is in and the state of the configuration item. Each role in the change process has a set of guidelines to determine authority for either approving change or escalating the change authority to the next level. This results in a compromise between a stringent, centralized change authority and the flexibility of rapid change required by an iterative process.

All changes are subject to the following and must follow a change request review cycle:

Development team lead approval and architect approval are required for change request of the Software Development Process specifications definition.

Development team leads have the authority to approve programming change requests but must defer architecture change requests to the project architects.

An architect has the authority to approve architecture change requests but must promote the change request to the CCB if a change to the architecture has budgetary, schedule, or quality ramifications.

The CCB has the ultimate approval authority for changes that impact budget, schedule, overall product functionality and quality.

Figure 2 illustrates the Change Management Process flow.

Figure 2: Change Management Process Flow

|  |
| --- |
|  |

Status reports can be generated at each step, and external processes, and the process audit are performed throughout the change management process.

When it has been determined a Configuration Item (CI) needs to be acted upon the users will submit a change request (1.1).

If necessary, the assigned personnel will triage (research the request) (1.2).

The CI then migrates through the SDLC, where it may need further analysis prior to the change being performed. (1.3).

Following CM published processes and procedures, the assignee will check out the artifact(s) to perform the change (1.4), and a new baseline is created (1.5).

After the above controls are in place, stakeholders can expect status reporting (1.6), described as recording, tracking, and reporting of information needed to effectively manage the project.

The quality of the CM processes can only be maintained, measured and improved upon by performing a process audit (1.6), which is an ongoing task that can be performed by either the CM team or an external entity such as SQA.

### CCB Review Process Flow

The CCB’s change review actions adhere to the model illustrated in Figure 3. This model serves as the structure of CCB roles and activities for each submitted CR or Issue.

Figure 3: Change Control Board Review Process

|  |
| --- |
|  |

### The Complete Change Control Process

The phases of Issues/Change Request and the activities of the CCB are combined into the complete Change Control Process *<PROJECT> CCB Charter* document.

### Issue Management

Any individual associated with the development project or executing its resulting products/application may identify issues. Issues may or may not result in a requirement for modification of the executing products/application or environment.

Issues that do result in such a requirement, will, by process, generate an associated Change Request(s) (either a Defect or Enhancement) in the Development Environment Change Tracking facility.

Issues that are solely related to project management issues such as scheduling or resource management concerns will follow the established methodology for issue resolution. Figure 4 displays the basic process flow for Issues within the Development Environment architecture.

Figure 4: Issue Process Flow

|  |
| --- |
|  |

### Change Management Request

Any individual associated with the development project may submit a Change Request within the development environment of the project. Change Requests may be submitted as a result of CCB Approved Issues requiring modifications to the developed products or as a result of the identification of needed modifications or enhancements while performing actions (such as development or testing) within the various project environments.

Figure 5 displays the basic process flow including both types (Defects and Enhancements) and conditions described above for Change Requests within the Development Environment architecture.

Figure 5: Change Request Flow

|  |
| --- |
|  |

The complete Change Request Flow is documented in the *Change Management* Processes and Procedures document, and may differ based on the tools used by the project team.

### Change Request Form

TBD

## Status Accounting

Configuration status accounting addresses the recording, tracking, and reporting of information needed to effectively manage the project. Data and logs that identify the configuration status, configuration item identification, and change status of all change requests must be maintained to enable reporting.

Change requests provide the foundation for configuration management status accounting with additional status reporting in support of this including: tool and resource usage, progress and the overall application’s CM Environment condition and state of readiness for backup and restore requirements.

### Status Reporting on Change Request Activity

The possible fields of a CR that may be reported on in a CR Status report either on a weekly basis or on an as needed basis are:

CR title (Change description)

CR priority (Low, Medium, High, Urgent)

CR Business Severity (Minimal, Medium, Severe)

CR state (Submitted, Approved, Rejected, …)

Only possible for CR State = Approved:

CR owner (Responsible engineer/assignee)

Projected Release Candidate date

Name of package impacted

Statement of problem

Action taken

### General Status Reports

Based on the status accounting tool, other reports can be produced as required. As stated above, other reports can include; tool and resource usage, progress and the overall application’s CM Environment condition and state of readiness for backup and restore requirements. The frequency, format and detail included in these reports are all determined by the needs of the project as required.

## Configuration Audits and Reviews

The audits and reviews are actions performed to verify the product matches the configuration items described in the specification and other documents and that the package is reviewed prior to release. The audits and reviews vary in complexity and formality. Generally, all products released will have gone through at the very minimum:

* A review by project management, or designee,for verification of functionality completeness. PM should review the list of functionality stated in the VDD to ensure it is correct.
* An audit by CM for accuracy by performing a reconciliation of the VDD against the actual objects and versions included in the release.

### Build Audits

Build audits consist of an as designed against an as-built verification, which is the comparison of the designed development of elements (technical documentation) versus the completed or built product. The CM staff will maintain audit trails of the builds that validate traceability of baseline changes, verify incorporation of approved changes, and provide integrity to baselines.

Audits are performed incrementally during development to validate the progress toward schedules and reduce efforts to successful testing. Unless otherwise directed, the CM Administrator will assume responsibility for verification that this activity is performed.

### Testing Audits

Testing Audits will be performed to assure quality and to assess the correctness, reliability, performance, robustness, and usability of the application by executing the application in a test environment with test data. Testers will follow the Release Management guidelines outlined in this document.

To have adequate configuration control of the testing environment, a formal test plans with all test cases to be run must be documented as well as the expected configuration, before any testing begins. When testing is complete, an as-run test plan along with all testing material is then archived in the CM Environment.

### Test Readiness Review

An informal test readiness review will take place directly before system testing to discuss any known deficiencies, impacting internal CRs, or test environment issues. The project lead, architects, developers, testers, CM Manager and the SQA Manager will attend the test readiness review, as well as any other employees who may benefit from the information.

## Release Management

A release is defined as the grouping of objects physical or electronic along with all the supporting media (documentation) necessary to satisfy a given set of requirements.

Release packages are created for a variety of reasons throughout the lifecycle of the project, the first of which is to allow for the initial grouping of CIs as they are formulated into a baseline. A release that has been initialized in the baseline stage of the project System Development Life Cycle (SDLC) can then be used as the medium for migration through all remaining stages of the SDLC up to and including release to production.

A release can be created to baseline existing production level objects for further enhancement. A release may be created if a product must be baselined for ongoing support while parallel concurrent development on the same initial product objects is initiated for updates to the release or subsequent releases.

The methods of packaging and rules concerning management of a release are determined by project management for the needs of the project along with consideration for the operational characteristics of the CM environment and its tools. For this project the primary form of a baseline will be in the form of an electronically packaged release. All project artifacts are subject to all the baselining and release management practices being used for this project.

### Release Packaging Concepts

A release is an object in itself that contains all the formally controlled objects that must be built, tested, and baselined and distributed as a single entity. Each object is managed by a component and a product group is likely to contain objects from more than one component, therefore, if you want to use the same objects in more than one version of a product, groups of objects are also arranged into releases. A release is a way of identifying the exact version of all objects that comprise a product at specific point in time.

A single version of an object may be included in several releases. Each time a development cycle begins for the next iteration of a product, a new release is then defined as well.

Each subsequent release of a product references many of the same objects as its predecessor. However, each release links to a particular version of each individual object. Thus maintenance of an older release may occur at the same time as development of a newer release.

### Electronic Release

The electronic release contains executable(s), installation scripts, user guide, installation documentation, and any required supporting software on electronic media. The executable is generated from the Baselined objects in the release component. The electronic production release media will be labeled with an identification number, descriptive name, and release date. A soft copy of the VDD will be generated and sent with the electronic product.

### Version Description Document

The Version Description Document (VDD) describes the documentation, media used to provide the release, and what is part of the release package (both physical and functional). The VDD contains the following:

Description of release media containing the electronic product identification and location of the packaged release.

Inventory listing all the configuration items used to make or generate the executable image. The list consists of the configuration item name, number, and revision level.

Functional and physical configuration audit reports.

Change descriptions containing one or both of the following:

Defect report listing defects that have been completed since the last release of the product and all open defects against this release of the product.

Scope Change Requests addressed by this release of the product along with a description of the changes made.

## The System Development Life Cycle

The System Development Life Cycle (SDLC) is used to identify and display all the stages that a product goes through during its evolution. All states or stages identified by documented development, test, packaging, and certification requirements are represented in the SDLC. The SDLC is not designed or engineered. It is simply the result of documenting what has been established by other activities in the Project and CM Planning stages. The main purpose and use for documenting the project SDLC is to map the project required stages and states to the terminology and configuration of the CM Tools implemented to support them.

The SDLC for <PROJECT> has multiple stages corresponding to the stages of development test and deployment of the releases. All tools in the development environment by default have been incorporated and configured to manage and control activity within the SDLC.

All packaged releases should be migrated through the SDLC by use of and adherence to the CM processes and procedures documented for this purpose. Figure 6 displays all the stages that may be used and the migration path through the SDLC.

Figure 6: SDLC and Migration Path

|  |
| --- |
|  |

# CM Environment

As stated, change control provides an integrated method for change initiation-processing, approval, and updating of a product baseline and/or release. The CM Review Board is responsible for ensuring the accuracy and integrity of the baselined repository.

The methods by which all activity within a CM Environment is conducted are determined by the functionality and configuration of the tools used for and within that environment. Figure 7 provides an overview of the components of a typical configuration management environment.

Figure 7: Configuration Management Environment

|  |
| --- |
|  |

The tools implemented for this CM Environment will provide the functionality depicted in above. This section provides information needed for actual <PROJECT> CM Environment configurations and naming conventions to be used.

## Development Tools

Various tools will be selected as the preferred development tool set for use in conjunction with the CM Environment tools for the <PROJECT> project.

## CM Tools

Formal SCM, Process, Document, and Change Management tools will be identified as the preferred version management, problem tracking, and notification tools for use within the CM Environment for the <PROJECT> Development Environment. All team members are to follow standardized instructions for using these tools in order to maintain consistency throughout the life cycle of the project. Instructional material including approved processes and procedures will be provided by and through the project CM Team as needed.

## <PROJECT> CM Environment Roadmap

This section contains a description of the structures available for use by the <PROJECT> project. The support requirements for multi-site, multi-platform development and test are the basis and major concern of this description. Included are methodologies to be followed pertaining to coordination of activities and deliverables from multiple distributed development and test locations.

## Requirements/Initial Conditions

This is a simple list of known conditions for SDLC activities conducted for this project, having requirements that must be supported by the formal CM Architecture.

* Distributed Development (multiple development sites possibly in multiple time zones).
* Consolidated Product Packaging and Release (single integration testing and single integrated product release).

## CM Architecture for the <PROJECT> Development Environment

Due to the locations chosen for development and testing of the products produced for this project, it will be required that a multiple independent site development methodology be introduced.

## CM Methodology – Distributed Development

This methodology has a high dependency on the modular architecture and design of the products to be produced.

|  |  |
| --- | --- |
| Advantages | Development Site Initialization and Configuration is greatly simplified. |
| Each site has operational independence to engineer and test their portion of the product, as they desire. |
| Each site repository need only be updated with stable and required versions of artifacts produced and maintained by other sites. |
| Changes are delivered to all sites only after completion of higher-level integration testing (later in the SDLC) reducing risk that a defect can be propagated project wide. |
| Maintenance activities and system outages have less potential for impacting activities at remote sites. |
| Disadvantages | Development Site Initialization and Configuration has to be performed. |
| Individual site environment administration must be considered. |
| Poor architecture (modularity) will increase complexity and degrade productivity. |

## Repositories

The SCM tool functions as an interface to a relational database, which will be maintained as the project’s primary archive repository. All CM Repository databases described in this section will be managed and controlled by formally documented and implemented CM processes and procedures designed to help ensure the most reliable and recoverable conditions are established and maintained.

### Status Accounting Repository

The status accounting repository stores all Development Change Requests which records and tracks all relevant events, episodes, or occurrences of change information that emerge during the course of the project’s development and maintenance. This repository will be used for the reporting of all current and historical data concerned with each CI throughout its life cycle.

### Development Environment Repository

The development repository stores all system files organized with regard to check in and checkout activity, security access, shared source code, history of changes, and development configurable item version control.

### Project Management Repository

The project management repositories store all Project Management Documentation, Project Issues, and Project Administrative objects. There will be a repository to house all other Project Management artifacts.

## Build Tools

The build utilities and their configuration have not been finalized at the time of this document’s publication; however, it is known that a process independent of the above listed tools will most likely be implemented. The recommendation for Java builds is Apache Ant (a Java-based build tool). The details of the build will be covered in the System Build Guide. Please contact a member of the CM team for information pertaining to the latest version and location of this document

### Component Builds

Component builds will be performed utilizing selected versions of the formally submitted source code contained within the Development Repository. Developers can and will build portions of the products within a private workspace area without affecting any other developer. However, to ensure consistency, repeatability, and especially product stability, the only official build is and will be the one conducted within the controlled environment by execution of the formal build process and procedures.

### IDE Integration

Development Tool IDE

Most of the commercially available development tools offer an interface for accessing source files that reside in the formal SCM repositories, however, with or without this capability, all official versions of the objects to be worked on must originate and be traceable to the formal SCM repository.

General processes and procedures for creation, checkout and check in of modifications are covered in the Change Management *Processes and Procedures* document. Contact the CM staff for information on the latest version and location of this document.

### IDE Configuration

Development Tool to SCM Tool Configuration

There will be formal tool integration between the developer IDE and the CM tool.

It will be up to each individual development team member to follow and use the published instructions.

## Guidelines for data organization

### Component Hierarchy

The component hierarchy is the structure that will be used to organize data and should reflect the organizational requirements of the developed products as well as the project organization and activities, and can be modified over time as those requirements change. Figure 8 illustrates a sample project component hierarchy.

Figure 8: Sample Project Component Hierarchy

|  |
| --- |
|  |

### Access Control

Access to work areas, change records, and components within the repositories for the <PROJECT> project is based on assigned tasks within a project. This practice is used to safeguard against inadvertent, unapproved, or unauthorized changes. The CM Administrator and Project Lead will manage access to each component residing in the project repository.

### Access List

Each repository will have controlled access based on the tools used to manage and interact with the repository. The access list manages user access (beyond the owner) to objects controlled within the repository by assigning users specific authorities. Access authority is inherited by lower-level components within each repository.

### Typical User Actions Within the CM Environment

As artifacts are created, they will be placed under SCM control. To change any existing artifact, it will need to be checked out from the SCM controlled environment, changed, and the new artifact, resulting from the changes made, will be checked back in, therefore updating the SCM archive repository.

The SCM tool stores additional information about the artifact (such as who changed it and why) each time an action is performed against it. This information can be queried at any time.

Common actions against artifacts, include:

Add to Source Control or Create or Submit: To store an artifact from your workstation or tool into the SCM repository, or to create or submit a record within the SCM repositories.

Check Out (Extract for Updating): To get a copy of an artifact onto your workstation for update.

Check In: To put an updated artifact back into the SCM repository creating a new version.

Extract (Read Only): To get a copy of an artifact onto your workstation without any intention of making changes to it.

Naming Convention:

Conventional SCM methodology requires that there always be unique names within the project’s versioned artifacts and releases.

When an artifact is created in the SCM environment, either just the **base name** is specified or the directory path in addition to the base name is specified. Specifying the **path name** as part of the name allows for several identical base named artifacts to be included in a single release.

Additionally identical artifact names within the context of a release can be created as long as their types and storage paths are different, such as file and data.

### Development Workspace (Work Area)

A workspace is an isolated work area where project team members perform their work. These are sometimes referred to as sandboxes or views of the CM repository. The team member checks out artifacts from the SCM repository, so that development and unit testing activities can be performed.

### Development Tool Updates

The development environment and the developer’s toolset must remain in sync, including any software purchased from an outside vendor. This can be controlled by way of a change request. These change requests include the authorization to introduce new tool releases into the development environment, which improves control points and prevents the installation of new tools that may negatively impact the development environment (i.e., a new compiler release just before a production release of the product).

Object-oriented development is dependent upon reuse and third-party class libraries. These items are also controlled and managed by the CM Staff as a complete library.

### Migration through the SDLC for the <PROJECT> Project

Project Development artifacts, Work Areas, QA Test and Release artifacts all pass through various stages within the SDLC for <PROJECT> as they evolve. Table 2 demonstrates the correlation between SDLC stages, associated CR Record States and Issue Record Status for a project fully managed by CM through to the Production environments. Variation from this mapping may be made depending on actual SDLC stages utilized for the project and possible delivery of products to environments prior to the production level stage depicted here.

CRs do not necessarily originate as a result of an issue submission. (See Issue and CR management sections in this document.)

Table 2: SDLC Stage: CR/Issue State Relationships

| Originating Issue Record State | Associated CR State | Development Stage |
| --- | --- | --- |
| Submitted | n/a | n/a |
| Approved | n/a | n/a |
| Rejected | n/a | n/a |
| Assigned | n/a | n/a |
| Resolved | Submitted | n/a |
| Closed | Submitted | n/a |
| Closed | CCB\_Rejected | n/a |
| Closed | CCB\_Approved | n/a |
| Closed | Approved | n/a |
| Closed | Assigned | DEV |
| Closed | Complete | FUNTST, INTBLD, INTTST |
| Closed | CM\_Complete | INTTST to SYSTST |
| Closed | QA\_Verified | SYSTST |
| Closed | UAT\_Verified | UATTST |
| Closed | Released | PRD |
| Closed | Deployed | PRD |
| Closed | Cancelled | n/a |
| Closed | Deferred | n/a |
| Closed | Resubmitted | n/a |
| Closed | Closed | n/a |
| Closed | Rejected | n/a |
| Closed | Request\_Info | DEV |
| Closed | Un\_Repeatable | INTTST |
| Closed | Postponed | n/a |
| Closed | Duplicated | n/a |

### Managing the Turnover Process

As previously described, the development team will turnover the configured items to the CM team after development unit testing has been satisfactorily completed. The Project Lead will inform the CM Manager that the development unit testing is successfully completed and is ready for the development integration-testing phase. After development high-level integration testing is completed, the turnover process will be managed in accordance with the <PROJECT> Migration Path and SDLC, discussed earlier.

### Parallel Development

Upon completion of an initial release of a software product, it is common to begin development of the next release. It is also common for defects to be discovered in the initial release that must be addressed immediately. This presents a conflict of the requirements for the software product. For example, one set of developers continues to develop new functionality; this may be called R1I2U0.

At the same time, defects are being discovered in test or release stages of R1I1U6. When defects requiring immediate attention are encountered, normally changes to the code are required. However, these defect fixes cannot include changes that are underway for the next release because of the risk of introducing additional errors or other implications of the new functionality. At this point the code must diverge into two independent lines of development called parallel development.

### Branching

Branching is the most common mechanism for controlling parallel development. Branches are deviations from the main development line (also called mainline) for an artifact. It allows the concurrent creation of versions and the splitting of development. To reduce complexity on the project, branching is limited at the release component level of the project. Branches only occur when a particular artifact must be updated for two distinct and different requirements (i.e. new functionality and defect fixes).

A new release will be created off of the main release containing the versions of the artifacts to be modified. If this is on a release other than the most current, a branch or new sub release is created for the purpose of fixing the defect and the new functionality development continues on the most recent or mainline release.

### Merging

After defect fixes are implemented, they are unit tested and follow the lifecycle of testing and promotion as described in previous sections of this document. The new release baseline must be formally accepted before it can be promoted and installed into production.

Merging refers to the procedure of analyzing and consolidating divergent pieces of software. Following the successful introduction of the defect fix into a release, the changes are merged into mainline development. Merging is performed after each successful defect resolution to reduce the risk of substantially different products. It is possible that the defect may have been resolved by the changes made for the new functionality. If so, analyzing the module changes on the branch and mainline highlights this outcome.

The owner of the mainline component is responsible for merging changes from the branched software into mainline development. Responsibility lies with the mainline developer because they have a better understanding of the functionality that currently exists in the mainline component, which may be substantially different than functionality that existed in the last release (and subsequent defect fix).

# CM Processes and Procedures

The documentation in this section contains the methodology; terminology and process flow to be used for <PROJECT>. For the documented processes and procedures to be executed in support of <PROJECT>, refer to the associated document for this project entitled Change Management Processes and Procedure document.

The Process and Procedures sections of that document are designed and formatted to allow for easy extraction and separate printing from the entire document. This is in an attempt to minimize the need to filter out the processes and/or procedures from all the associated front and end matter in the entire document for those who only need either of those sections to work with as a reference. Contact the CM Staff for the most recent version and location of this document.

## The CM Process

Each master process to be documented and executed for a project has five basic components (Sub-Tasks, Entry Criteria, Input(s), Exit Criteria, and Output(s)). Each subtasks maybe a master process in itself and thus described separately.

The master process is initiated when a proper entry criterion is perceived for any of the listed sub-tasks. The Point of Contact (POC) for each process will verify that all required inputs are available and upon completion of each sub-task in the execution, that all required outputs have been generated and all the necessary exit criteria has been met.

The master process is not considered completed until all related sub-tasks for the perceived entry criteria have been completed. Verification and validation is done to ensure that the process(es) executed meet(s) the needs of the development effort and that execution of this process satisfies the certification requirements of the organization requesting the activity.

### The CM Compliant Process Defined

After the production platform and product information is received, a *Basic CM Compliant* process based on the requirements of each product can then be defined. The main focus of such a process would be to establish and document the current requirements then begin to establish the steps necessary to apply CM Compliant tools, processes and procedures as needed in support of the established requirements.

The basic steps needed for the <PROJECT> CM Compliant process, include:

**Entry Criteria** - What events have occurred that make execution of this process required.

1. Example: A product enhancement/update request and/or directive have been issued.

**Inputs** - What components are required to begin execution of this process.

1. Example: Tools, personnel, software, documented procedures to be executed such as a migration procedure, or a compile procedure are generally listed here.

**Exit Criteria** - What procedures have to be completed to satisfy completion of this process.

1. Example: The compile procedure has been completed and notification has been published.

**Outputs** - What are the products of this process.

1. Example: The executables produced by a compile process and the notification that was sent.

**Verification and Validation** - What is done to guarantee that this process is the correct process to be executed and that it was executed successfully.

1. Example: An Audit process is normally triggered on exit from a process that describes where and how periodic inspection and review of the outputs should be done.

A process with these basic steps documented provides CM with the ability to support a formally structured Source Code Management tool. A formally structured SCM tool helps guarantee the ability to back out a change, as well as provide a more stable and structured environment in which to perform concurrent development and testing. Accountability and traceability features are also available for all versioned code.

### CM Process Flow Model

Figure 9 illustrates the typical CM process flow model that can be used for all projects that produce software to be brought into CM Compliance.

Figure 9: CM Process Flow Model

|  |
| --- |
|  |

When it has been determined a Configuration Item (CI) needs to be acted upon the users will check out the CI for performing the update to it (1.1) out of the determined repository following the published processes and procedures.

After the user completes the action on the CI that was checked out the next step will be for the user to check in the CI (1.2) into the determined repository following the published processes and procedures.

The CI then migrates through the SDLC (1.3), used to identify and display all the stages that a product goes through during its evolution.

After the above controls are in place, stakeholders can expect status reporting (1.4), described as recording, tracking, and reporting of information needed to effectively manage the project.

The Backup and Recovery (1.5), piece of the CM process is an ongoing initiative for making sure that all the assets of the projects are protected and can be recovered per the documented process and procedures.

The quality of the CM processes can only be maintained, measured and improved upon by performing a process audit (1.6), which is an ongoing task that can be performed by either the CM team or another entity such as SQA.

### Backup and Disaster Recovery Plans

Recovery of modifications to Project Configurable Items is possible at multiple levels within the CM environment tools, depending on the nature of changes to be recovered and the activities leading up to a need for recovery.

Recovery of the CM environment itself is supported by the currently implemented process for backup and recovery and will be used for recovery of data depending on the circumstances requiring a restoration.

The Veterans Health Administration’s Tier-One Hines Regional Processing Center (HRPC) will perform the recovery process in coordination with HSD&D and/or their designated personnel. Data loss will be assessed by the HRPC and the <PROJECT> team will determine what files need to be recovered. At that point, the HRPC will begin the recovery process. After the files are restored, the HRPC will ensure the recovered application is operational. Once that is determined, HSD&D or the designated personnel will conduct system and database tests to ensure the application has returned to normal operating status. An appointed <PROJECT> project member(s) will verify within 2 hours whether the application has been returned to normal operating status.

The procedure’s and specific details related to backups performed by the HRPC are located in the “Veterans Health Administration, Tier 1 Regional Processing Center, Performance Service Level Agreement (PSLA)”, Section 1.01, for Rational Enterprise Suite.

The procedure for requesting a restore of any *<PROJECT>* Configurable Item is located in the *Restore of <PROJECT> CIs from Backup* in the *CM Procedures* section of the Change Management *Processes and Procedures* document.

### Method for CM Environment Configuration Recovery

All system configuration files for the <PROJECT> CM Environment are maintained in a separate location from the CM tools used for the Project Configurable Item Archives. In the event of a corruption of the CM Environment architecture itself, recovery will be attempted by one of the following two methods.

If corruption of the environment occurs based on inadvertent deletion or modification of the configuration files, the first method of recovery to be attempted will be with CM backups of the configuration files maintained on the server. The most recent revision of the corrupted configuration files is retrieved out of the CM Environment archives and used to restore the damaged area.

In the event that corruption extends beyond the ability to restore from the zip files maintained on the server, a restore would be from the latest weekly server backup performed by the (HRPC) group.

### Method for <PROJECT> Site Servers Recovery

The HRPC maintains a schedule of nightly partial backups and weekly full backups according to the “Veterans Health Administration, Tier 1 Regional Processing Center, Performance Service Level Agreement (PSLA),” Section 1.01, for Rational Enterprise Suite.

These servers contain the entire installation of <PROJECT> archives and all related CM Environment configuration files. If a loss of data is experienced to any or all of these servers, the CM team can be contacted to coordinate the restore all required directories and/or files from the most recent backup containing the uncorrupted data.

### Weekly HRPC Backup and Recovery

A more comprehensive and thorough backup process for all environment servers will be conducted according to the Veterans Health Administration, Tier 1 Regional Processing Center, Performance Service Level Agreement (PSLA), “Section 1.01, for Rational Enterprise Suite*.*” This document also governs the recovery of a small scale (one file) up to the larger scale of an entire server.

### Communication and Environment Coordination

Various forms of communication and coordination of activities will be utilized within this project, and may be used in conjunction with email. After the formal processes are in place, certain actions that might be presently conducted through email should be conducted through formal Project Management utilities designed for distribution of information. Email will be used as a backend to the Utilities operating in parallel.

# Acronyms, Abbreviations, and Terms

## Acronyms and Abbreviations

| Acronym | Definition |
| --- | --- |
| CCB | Change Control Board |
| CM | Configuration Management |
| COTS | Commercial-off-the-shelf products |
| CMP | Configuration Management Plan |
| ECCB | Environment Change Control Board |
| EVS | Enterprise VistA Support |
| HRPC | Hines Regional Processing Center |
| HSD&D | Health System Design & Development |
| ICCB | Internal Change Control Board |
| IDE | Integrated Development Environment |
| IDL | Iterative Development Lifecycle |
| IV&V | Independent Verification and Validation |
| JAD | Joint Application Development |
| NSR | New Service Request |
| POC | Point of Contact |
| QA | Quality Analyst |
| RAD | Rapid Application Development |
| SCM | Software Configuration Management |
| SDLC | Software Development Life Cycle |
| SQA | Software Quality Assurance |
| SRS | Software Requirement Specifications |
| TCCB | Technical Change Control Board |
| TSPR | Technical Services Project Repository |
| UCM | Unified Change Management |
| UML | Unified Modeling Language |
| VDD | Version Description Document |
| VHA | Veterans Health Administration |
| **V***IST***A** | Veterans Health Information System and Technology Architecture |
| VOB | Versioned objects-based |
| WPR | Work Product Review |

## Terms

| Term | Description |
| --- | --- |
| Architecture | The organizational structure of a system or component. (IEEE Glossary 1990)  The architecture of a software system (at a given point in time) is its organization or structure of significant components interaction through interfaces, those components being composed of successively small components and interfaces. (RUP 5.5 1999) |
| Audit | In CMMI process improvement work, an independent examination of a work product or set of work products to determine whether the requirements are being met. (MMI-SE/SW, v1.02)  The process of conducting an official examination as a means to evaluate quality and integrity against predefined standards. (ITIL IT Service Management Essentials, v6.0) |
| Baseline | A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and can be changed only through formal change control procedures.  A document or a set of such documents formally designated and fixed at a specific time during the lifecycle of a configuration item. (IEEE Std. 610.12-1990) |
| Build | An operational version of a system or part of a system that demonstrates a subset of the capabilities to be provided in the final product. (RUP 2003.06.12.01 2004) |
| Change Management | Process of controlling changes to the infrastructure or any aspect of services, in a controlled manner, enabling approved changes with minimum disruption. (ITIL IT Service Management Essentials, v6.0)  Judicious use of means to effect a change, or proposed change, on a product, or service. (See also “Configuration Management) (CMMI-SE/SW, v1.02) |
| Change Request (CR) | A general term for any request from a stakeholder to change an artifact or process. Documented in the change request is information on the origin and impact of the current problem, the proposed solution, and its cost. (RUP 5.5 1999) |
| Change Control Board (CCB) | Sometimes known as Change Control Board. A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items and for ensuring implementation of approved changes. (IEEE Std. 610.12-1990) |
| Configuration Item | An item that is identified to be placed under the control of configuration management. (CMMI-SE/SW, v1.02) |
| Configuration Identification | An element of configuration management, consisting of selecting the configuration items for a product, assigning unique identifiers to them, and recording their functional and physical characteristics in technical documentation. (CMMI-SE/SW, v1.02) |
| Configuration Status Accounting | An element for configuration management, consisting of the recording and reporting of information needed to manage a configuration effectively. This information included a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of approved changes (see also “configuration management” and configuration items” for contrast.) (CMMI-SE/SW, v1.02) |
| Configuration Management | A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. (IEEE Std. 610.12-1990)  The five major activities of configuration management are:  Configuration Identification  Configuration Control  Configuration Status Accounting  Configuration Audit  Delivery. |
| COTS | Commercial off the Shelf (CMMI-SE/SW, v1.02) |
| Delivery | Release of a system or component to its customer or intended user. (IEEE Std. 610.12-1990) |
| Product | A product is a “work product” that is delivered to the customer. (CMMI-SE/SW, v1.02) |
| Product Life Cycle | The period of time that begins when a product is conceived and ends when the product is no longer available for use. (CMMI-SE/SW, v1.02) |
| Project | A managed set of interrelated resources that deliver one or more products to a customer or end user. This set of resources has a definite beginning and end typically operated according to a plan. (CMMI-SE/SW, v1.02) |
| Project Manager | The person responsible, directing, controlling, structuring, and motivating the project. (CMMI-SE/SW, v1.02) |
| Release | A subset of the end product that is the object of evaluation at a major milestone. A release is a stable, executable version of product, together with any artifacts necessary to use this release, such as release notes or installation instructions. A release can be internal or external.  An internal release is used only by the development organization, as part of a milestone, or for a demonstration to users or customers. An external release (or delivery) is delivered to end-users. A release is not necessarily a complete product, but can just be one step along the way, with its usefulness measured only from an engineering perspective. (RUP 2003.06.12.01 2004) |
| System | An integrated composite that consists of one or more of the processes, hardware, software, facilities and people, that provides a capability to satisfy a stated need or objective. (ITIL IT Service Management Essentials, v6.0) |
| System Development Lifecycle (SDLC) | A conceptual model, used in [project management](http://searchVB.techtarget.com/sDefinition/0,,sid8_gci951200,00.html), that describes the stages involved in an information software development project, from an initial feasibility study through maintenance of the completed application. Various SDLC methodologies have been developed to guide the processes involved, including (but not limited to) the Iterative development life cycle (IDL); the Waterfall model; rapid application development ([RAD](http://searchSmallBizIT.techtarget.com/sDefinition/0,,sid44_gci214246,00.html)); and joint application development ([JAD](http://searchSmallBizIT.techtarget.com/sDefinition/0,,sid44_gci820966,00.html)). |
| Traceability | The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match. (IEEE Std. 610.12-1990) |
| Version | An initial release or re-release of a software product, associated with a complete compilation or recompilation of the software product.  An initial release or re-release of a document. (IEEE Std. 610.12-1990) |
| Work Product | Any artifact produced by a process. (CMMI-SE/SW, v1.02) |

* 1. Attachment A Approval Signatures

This section is used to document the approval of the Configuration Management Plan during the Formal Review. 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.

//es// Kelli Montali\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

Kelli Montali

Director, Process Management Service

NOTE: Delete the entire section above prior to final submission.

REVIEW DATE: *<date>*

SCRIBE: *<name>*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Program/Project Manager >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Business Subject Matter Experts Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Software Engineering Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Software Development Manager >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Information Protection Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Section 508 Compliance Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Enterprise Infrastructure Engineering (EIE) Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Field Operations and Development (FOD) Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Independent Verification and Validation Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Operational Readiness Testing Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Release Management Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Product Support Representative >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Implementation Manager >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Planner >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Training Manager >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< Contracting Officer >*

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*< General Counsel >*