USAID | DELIVER PROJECT

HEALTH COMMODITY MANAGEMENT INFORMATION SYSTEM (HCMIS)

CONFIGURATION MANAGEMENT PLAN

FOR HUB EDITION

JULY 2011

This publication was produced for review by the U.S. Agency for International Development. It was prepared by the USAID | DELIVER PROJECT, Task Order 4.

HEALTH COMMODITY MANAGEMENT INFORMATION SYSTEM (HCMIS)

CONFIGURATION MANAGEMENT PLAN

FOR HUB EDITION

USAID | DELIVER PROJECT, Task Order 4

USAID | DELIVER PROJECT, Task Order 4 The USAID | DELIVER PROJECT, Task Order 4, is funded by the U.S. Agency for International Development (USAID) under contract number GPO-I-00-06-00007-00, order number AID-OAA-TO-10-00064, beginning September 30, 2010. Task Order 4 is implemented by John Snow, Inc., in collaboration with Asociación Benéfica PRISMA; Cargo Management Logistics; Crown Agents USA, Inc.; Eastern and Southern African Management Institute; FHI 360; Futures Institute for Development, LLC; LLamasoft, Inc; The Manoff Group, Inc.; OPS MEND, LLC; PATH; PHD International (a division of the RTT Group); and VillageReach. The project improves essential health commodity supply chains by strengthening logistics management information systems, streamlining distribution systems, identifying financial resources for procurement and supply chain operation, and enhancing forecasting and procurement planning. The project encourages policymakers and donors to support logistics as a critical factor in the overall success of their healthcare mandates.

Recommended Citation

USAID | DELIVER PROJECT, Task Order 4. 2010. Health Commodity Management Information System (HCMIS) Configuration Management Plan for Hub Edition. Arlington, Va.: USAID | DELIVER PROJECT, Task Order 4

USAID | DELIVER PROJECT

John Snow, Inc.

1616 Fort Myer Drive, 11th Floor

Arlington, VA 22209 USA

Phone: 703-528-7474

Fax: 703-528-7480

E-mail: askdeliver@jsi.com

Internet: deliver.jsi.com

CONTENTS

1 INTRODUCTION

Configuration Management Plan (CMP) describes the plan for tracking the configuration of elements of the developed system and managing changes throughout the lifecycle of the HCMIS HE project. HCMIS HE is a systematic record-keeping system, which can help efficiently manage daily transactions at warehouses or facilities. The Software Configuration Management (SCM) will be used to manage and control versions, and provide change history for the software product deliverables and project documentation.

This document, therefore, will be revised and republished to remain consistent with other project documentations such as Detailed Functional Requirement, Test Plan, and Risk Management Plan documents.

1.1 Purpose

This CMP describes the Change Management (CM) responsibilities and processes that support the design and implementation of the HCMIS HE. The purpose of this CMP is to identify the organization providing the configuration control, define what a configuration-controlled item is, describe the change control process, and identify the plan for configuration status accounting and verification.

This CMP is designed to ensure that:

- Baselines are defined and documented
- Documentation is identified, released and controlled
- The Configuration Management Team (CMT) is established and functions according to CMP guidelines
- Changes to the baseline are evaluated and controlled
- Approved configuration changes are implemented and tracked
- Configuration status accounting is accomplished

1.2 Scope

This CMP is applicable to all work performed as part of the HCMIS HE project, which includes the design, integration and test the project. It provides guidance for all personnel on CM activities in support of the HCMIS HE project, including all development and project team. CM is applied to

items selected by the Project Manager and specifications, procedures and other support documents. The scope of this CMP encompasses the lifecycle of the HCMIS HE Project.

1.3 Audience

The intended audience for this document includes individual such as developers, testers, project sponsors, project manager, and project team members who need access to or work within the HMIS HE. In addition to this, the CMT must understand and comply with the procedures set forth in this document.

In general, all project participants are responsible for reading and understanding the content of this document. HCMIS HE personnel and Project Management (PM) will use the procedures in this document to manage the implementation of the configuration as well as the project library.

1.4 Constraints and Assumptions

A list of constraints and assumptions is provided below, as they impact the cost, schedule, scope, content, development approach, team, review, acceptance, or ability to perform configuration management activities.

- The schedule of CM activities and their task dependencies will be achieved
- Delivery of work products will be provided within a sufficient time frame to allow required CM activities to be performed.
- Interface with other organizations, e.g., Ministry of Health and the Pharmaceutical Fund & Supply Agency (PFSA).
- This document covers the software Configuration Management and documentation in association with or used in conjunction with the HCMIS HE
- When the HCMIS HE matures, it will be installed at different Hubs (warehouse). Through the course of the development process, this document may be modified to capture any Configuration Management process modifications that are made by the project management team
- The Configuration Management is a subset of the development standards and processes for the HCMIS HE

1.5 Approach

For managing and controlling changes to HCMIS HE and the associated deliverable documentation, CMP provides an effective strategy. The HCMIS HE CM manager is responsible for executing the plan that institutes the following key components:

- CM Planning
- Configuration Identification
- Change control
- Deployment
- Configuration Status Report
- Audit and Reviews
- Documentation

2 ORGANIZATION

The HCMIS CM activities are handled by the following groups:

- Project Team Leader
- IT Analyst
- Development Team
- Testing Team
- Documentation Consultant
- Training Coordinator
- Configuration Management Team
- Quality Assurance Team
- Risk Management Team

The HCMIS HE CM group consists of the CM manager and CM team members (for further description of CM roles and responsibilities, see Section 2.3).

In addition to this, the CM group will interface with other groups including the project team, IT Analyst, and development team as necessary.

2.1 Structure

The CM organization is illustrated in Figure 1. In the HCMIS HE CM organization, the CM manager oversees the CM team, all CM activities and issues. Moreover, the CM manager reports CM activities and issues to the Project Team Leader (PTL).

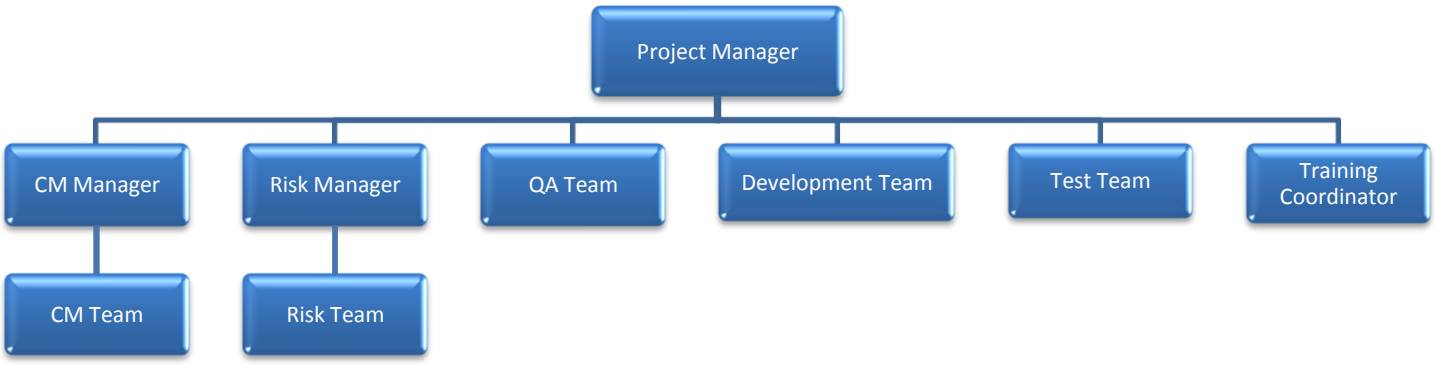

Figure 1: Configuration Management Team Structure

2.2 Authorization

CM takes direction for the PTL and operates within the policies and procedures established by USAID | DELIVER PROJECT. Configuration Management is organized to utilize by USAID | DELIVER PEOJECT to establish CM procedures, oversee the application of these procedures by all team members and provide all necessary reports and support for the CM functions. In addition to this, Configuration Item (CIs) covered by this plan, affect the project as a whole, its stated purpose, development, and CM team to maintain the quality, availability, and integrity of the IT infrastructure. Hence, it is the responsibility of all stakeholders of the CM to successful implementation of the Configuration Management procedures.

2.3 Roles and Responsibilities

The primary CM responsibilities for each of the CM organizational roles, as identified in Section 2.1, are discussed in this section and are provided in Table 1. These responsibilities are defined for the entirety of the HCMIS HE lifecycle.

Table 1: CM Organizational roles and responsibilities

Role	Responsibilities	Organization
Project Team Leader	• Ensures compliance with HCMIS HE processes, procedures and conventions • Provides oversight of CM activities to ensure successful program performances and compliances with program policies • Ensures adequate resources are available for CM activities • Ensures that support team leads and other persons in management or supervisory roles support the objective of this CMP • Resolves open issues by making final decisions	Project Management

Role	Responsibilities	Organization
Development Team Leader	• Manages development tools • Ensures that development artifacts (source code and design documents etc.) are under revision control • Ensures the integrity of the software revision control • Communicates with the development team and problem resolution	Development Team
Testing Team	• Manages test tools • Ensures that testing artifacts are under revision control • Verifies change requests implemented in release build • Responsible for developing test criteria and plan with the development team	Testing Team
IT Analyst	• Manages HCMIS HE tools • Ensures the integrity of release Configurations • Communicates with the team that authorizes builds to the environment • Closely works with the project team on test plan development and training related issues • Actively participates in project plan development, scope development and issue resolution • Works with the rest of the project team so as to ensure that project deliverables are met	Project Management Team
Change Management	• The CM Leader, designated by the Project	CM Management

Role	Responsibilities	Organization
(CM) Leader	Manager (PM), is responsible for executing and directing CM activities • Closely works with the Project Manager (PM) to ensure that adequate resources and funding are available to perform CM activities • Establishes the CM environment • Ensures CM Team receives adequate training to perform CM activities • Prepares and submits standard reports to document the status of CM activities to the Project Manager (PM) • Reviews and monitors the development team's CM activities • Identifies opportunities for CM processes	
Change Management (CM) Team	• Establishes the HCMIS HE Configuration Management Library (CML) • Assists with the administration and management of CM tools and the management of the CM environment • Maintains and documents system configuration items and baselines • Records and reports configuration history for CM items	CM Management
Documentation Head	• Manages document change processes • Ensures that documentation is under revision control	Documentation Consultant
Configuration Quality	• Manages baseline audit processes	Quality Assurance

Role	Responsibilities	Organization
Assurance (QA) Leader	• Reports audit results to Project Management Team • Ensures the integrity of the release baselines	(QA) Team

3 RESOURCES, TRAINING AND SCHEDULE

The following sections describe the staffing (personnel), training, schedule and other additional things that are required to implement CM activities for the HCMIS HE project.

3.1 Staffing

The following table summarizes the number of staffs with their position and required skill for the CMP.

Table 2: Number of staff, position and required skill for Configuration Management Plan

Number of staff	Position	Required skill
1	Project Team Leader	3-5 years of experience in project management area
1	CM Manager	2-4 years of experience in software development
3-5	CM Staff	Minimum of 2 years of experience in IT
1	Risk Manager	3-5 years of experience in project management
2-3	Risk Management Staff	Minimum of 2 years of in project management
2-3	Quality Assurance Staff	Minimum of 2 years of in QA
2	Developer	Minimum of 3 years of experience in software development and database design
2-4	System Analyst	Minimum of 3 years of experience in software engineering
3-5	Tester	1-2 years of experience in software testing
2	Documentation Consultant	Minimum of 2 years experience in software engineering
1	Training Coordinator	2-4 years of experience in software testing

3.2 Training

CM Team requires an essential proper training in order to accomplish their tasks efficiently. The required training needs are determined by matching skill requirements for specific task against the skills of the assigned personnel. Specific training needs to support CM activities are listed below.

- Users training on CM processes and procedures
- User training on the HCMIS HE procedures for CM
- Configuration control process training

3.3 Schedule

This section defined the schedule by splitting the modules into several tasks and specified the schedule according to their deadlines.

Table 3: Schedule

Activities	Target Date
First development snapshot demo	DATEMISSING
Pre-first release prototype	DATEMISSING
First release	DATEMISSING
Final release	DATEMISSING

4 STANDARDS, PROCEDURES AND POLICIES

4.1 Standards

4.1.1 Language

Every function, variable, class, and file naming and documentation in source code must be in English.

4.1.2 Naming

Standardized naming schemes must be followed to promote uniformity and clarity across the product development lifecycle.

4.1.2.1 Source file naming convention

The first letter of each word in the filename must start with a capital letter. The name can not contain any spaces; however, it may contain underscores (_). The name must be descriptive for its functionality, but may not exceed 32 characters in length.

4.1.2.2 Function naming convention

Each function name must start with a lower case letter using capital letters to separate words. It cannot contain an underscore (_). The name must be descriptive of its functionality.

Functions that set a member variable must start with "set" in the name. Similarly, functions that retrieve a member variable must start with "get" in the name.

4.1.2.3 Variable naming convention

All locally accessible variables must start with a lower case letter and use capital letters to separate words. It cannot contain an underscore (_). All globally accessible variables must start with the word "global" and following the same standard as locally accessible variables. All constants must contain only capital letters and may use underscores to separate words.

4.1.2.4 Tags

Tag names and their attributes/options may only contain lower case letters and cannot contain underscores (_).

4.2 Semantics

4.2.1 Declaration and Assignment of Variables

All variables must be explicitly defined and initialized at the beginning of a code block. Strings must at least be initialized to the empty string (“”); numbers set to zero; and Booleans set explicitly to true or false.

4.2.2 Return Statements

There can only be one return statement per code block to mark clearly when a value is returned.

4.2.3 Tag-based Languages

When using a tag-based language, both beginning and end tags must be included, unless intentionally exempted by the language.

4.2.4 Comments

All functions must be prefaced with a brief description of its purpose, the expected parameters, and its return value. Critical and dense blocks of code should be prefaced with a brief purpose and include periodic notes within each sub-block.

4.2.5 Source Code Layout

Any text editor is acceptable; however, indentations may only be made with spaces to ensure compatibility across all editors. Some editors offer “soft tabbing” which converts tabs into spaces while saving, but no indentation should exceed four spaces beyond its parent level.

4.3 Procedures Overview

Table 4 documents the procedures that are involved in the change management process of the HCMIS HE.

Table 4: CM Procedures

Procedure	Short Description
Change control	Controlling changes that will be made in developing HCMIS HE
Revision Control	Ensuring changes to items do not get lost or overwritten in developing HCMIS HE
Version Control	Controlling various version of documents and software products for HCMIS HE project
Product Builds/Release Control	Ensuring released version of HCMIS HE products have some kind of format
Status Reporting	Prepare and provide appropriate configuration status report for decision making process
Data Recovery and Backup	Procedures used to keep backup and recovery during in developing HCMIS HE
Release resource	Ensure how internal and external releases are maintained and controlled

4.4 Software Configuration Management (SCM) Polices

The establishment of Configuration Management is essential for the HCMIS HE in order to:

- understand when a work product has been baselined
- ensure that no unauthorized requirement, design, or code change is made after a baseline has been established
- ensure that no configuration item is changed by more than one engineer at any time
- ensure that the impact of any change on a configuration item is evaluated, understood and managed
- capture the current state of the product at any given time

5 SOFTWARE CONFIGURATION MANAGEMENT (SCM) ENVIRONMENT

HCMIS HE environment scope is defined based on the content and implementation required to support the HCMIS HE standards, procedures and policies.

5.1 Contents

Allowing access to all of the elements is one of the purposes of revision control in the repository.

Besides, revision control plays a key role by ensuring that all changes that are introduced though the course of the development process do not get lost or overwritten. Revision history for all elements should be able to view by users of the revision control system, compare the differences between revisions, and, at times, merge revisions together to form new revisions. They should also be able to retrieve already existing versions.

The following listed components are to be put under revision and ultimately baseline control:

- Source Code
- Design artifacts
- Documentation/specifications
- Test artifacts
- Project library items (plans etc.)

Table 5: Configuration item, element and extension

Configuration Item	Element	Extension
Source Code	C# source file	*.cs
	.Net Project file	*.sln
	SQL Script	*.sql
	Image File	*.jpg
	Image File	*.gif
	Image File	*.png
	SQL Server Index File	*.ind
	SQL Server Stored Procedure Body File	*.spb

	SQL Server Stored Procedure Definition File	*.spd
	SQL Server Table Definition File	*.mdf
	Build Script	*.xml
Architecture	Design document	*.doc
Test Artifacts	Test Suite	*.xsl
Test tracking	Metrics	*.xls
Core Internal Documentation	Project Schedule (MS Project)	*.mpp
	Test Plan	*.doc
	Software Configuration Management Plan	*.doc
	Business Requirements Specifications	*.doc
	System Requirements Specifications	*.doc
	Detailed Design Document	*.doc
User Documentation	User Documentation	*.doc

5.2 Attributes

Name, identifier, identity of the person who initially created the item, date/time the item was initially created, revision number and history of the revision are attributes that the item will be maintained the items in the repository.

For each revision, the identity of the person who applied the version, date/time the revision was made and revision description (as entered at the time of the revision - may include CR number) with attributes included.

In order to retrieve any version as well as all the version attributes, the history of the revision will be used.

5.3 Identification and Traceability

Configuration Management extends to documents as well as project source code. Unique identifier will be assigned for each document that will be configured separately. When it comes to the project source code, it might be broken down into separately configured pieces as necessary. For instance, routines that are useful apart from the main body of the code should be split into their own separately configured library. This separation could make the work more useful to the development process.

Each CI (document, source code or other) will be assigned a short mnemonic identifier (i.e. CI name) that will be used by the Configuration Management Database (CMDB) system to refer to that particular CI.

A list of all CIs, a short summary/description of the CI, and a "Lead Developer" for the CI will be maintained by the Configuration Manager (CM) and it will be well documented.

5.4 Release Identifier/Software Version Structure

5.4.1 Release Definition

The following convention will be used as a guideline for software release numbers:

- **Major Release** – a release that contains significant new functionality or technology based enhancement. Most major releases are anticipated to require a complete installation procedure and may require data migration/conversion. Significant new test must be developed and full regression test will be required.
- **Minor Release** – a release that contains a new feature or technology enhancement. Minor releases will typically be applied as an upgrade rather than a complete reinstallation. Some new test must be developed and minimally a partial multi-unit test will be required. Revised product documentation and modification to training will be required.

5.4.2 Format

A release identifier follows this format **MM.mm.ii.pp.bb[aaa].**

Where:

MM is the major release number

mm is the minor release number

ii is the maintenance release number

pp is the patch release number

bb is the build number

aaa is an optional alpha field

Fields are changed by incrementing, i.e., there should be no gaps in the numbering. MM, mm, pp, ii and bb are one or two digit numeric with optional leading zeros. aaa is a 1-3 character alpha only field. Each software version should have a unique release identifier. Typically, for a given release there will be several software deliveries (SCM builds) delivered to test. The fifth field tracks the software build. The build number is reset to one whenever the version number (MM, mm, ii or pp) changes.

5.4.3 Release Media and Format

In releasing the system CD-ROM is used for the first time, Updates, maintenance and patch releases will be made available via a secure server access.

5.5 Software and Hardware Environment

5.5.1 Tools

Throughout the course of developing the HCMIS HE system, an open source web-based system (i.e. BugNet) is put in place as a major Configuration Management tool for tracking changes.

5.5.2 Repository

The below lists are repositories that HCMIS HE uses for the storing documents and source code. These repositories are located on a local server.

5.5.2.1 Document Repository

The document repository is made up of a database that references each file

5.5.2.2 Code Repository

The code repository will contain all the files used by the system as organized using the .NET folder styling.

6 PROCEDURES

6.1 Change Control Procedures

To request changes on the HCMIS HE with all changes under change control the client or project team requires a Change Requests (CRs) to be opened and its progress tracked until verified by a successful test and closed.

To review the severity and determine the priority of all non-closed CRs, the CM team will meet periodically. Changes to be included in the next release build will be determined and authorized by the CM manager. The release configuration used for the build will be baselined with a unique release/build identifier and will be associated with the authorized CRs.

6.2 Revision Control Procedures

Revision control able to maintain changes to items do not get lost or overwritten and provides access to the entire unit in the repository. Stakeholders of the system are able to see the revision history of all units and compare the differences between revisions.

The following sections explain the revision control procedures to be used during the various test stages.

6.2.1 Multi-Unit, System and Acceptance Tests

Developers will be using a controlled reserved check-in/check-out approach to source control during the multi-unit, system, and acceptance test state. Only one developer will be able to check out a file at a time. Each workspace will be a snapshot view that points to the baseline the CR was reported against, isolating the specific CR task from the rest of development.

Table 6: Multi-Unit, System and Acceptance Tests

Step	Role	Action
1	Developer	Creates snapshot view from the current version with login id and CR number in name (see Workspace Naming Conventions)
2	Developer	Checks out code with "-reserved" option
3	Developer	Modify code, build, unit test, verify
4	Developer	Wait for the lead developer to call for view name

6.3 Version Control Procedures

0.1 is the starting point for all documents as initial versioning standard. When minor changes are made the new version will be incremented by 0.1, for example changes made to the initial version will be 0.2, then 0.3, etc. For all documents to be versioned 1.0 or 2.0, they must be agreed by the project team and signed off by both parties.

If more than one person makes changes to a document than the versioning will appear as 1.1.1 for one person and 1.1.2 for another. When both are combined, the version will appear as 1.2 etc.

6.4 Product Builds/Release Procedures

In this section of the Configuration Management Plan and procedure document describes the procedures for product builds/release

6.4.1 Multi-Unit, System and Acceptance Test

The HCMIS HE system developer being informed that all code has been checked in by the lead developer, the HCMIS HE system developer will create a snapshot view that includes the build number I the name and labels the tip revision with the requested release and build number.

If the build is successful, the final steps should be the automatic copy of the deliverable products to a release area and a compressed archive file generated.

Table 7: Product Builds/Release Procedures Specification

Release Media	Release Directory	Archived Elements	Ext
Aaa_<<version>>.zip	aaa\bin\Debug	Executable files	.exe

The build product release tar file will follow an <<appname>>_<<version number>>.zip naming convention (i.e. aaa_1.1.zip).

Convention: <<appname>>_<<version number>>.zip

<<appname>>	Project acronym (i.e. aaa)
<<version_number>>	Version number (i.e. 1.1)

6.4.2 Release to Field

The final release configuration, application, and product documentation will be handed off to project management for final packaging and release to the end users once testing has completed

Table 8: Releases, Deliverables and Extensions

Release Media	Deliverable Components	File Extension
Master Application on CD-ROM	Archived build product	*.zip
Soft Copy	Release Letter	*.doc
Documentation Soft Copy	Product documentation	*.doc

6.5 Status Reporting Procedures

To prepare and maintain appropriate Configuration Status Report (CSR) for the members of the project team the CM tracker system will be used. Certain information will be stored in the CM tracker system for easy access by project team. Other additional reports will be prepared by the SCM organization and distributed.

Adequate CSR will be maintained to support the delivery and acceptance of the system. The CSR will track the status of all configuration documentation, software elements, and all configuration changes actions to ensure that the current approved configuration is always readily known. The CR activity will be tracked and its status maintained. The following applicable elements will be included in the reports:

- Document number, including revision number
- Configuration Identification (CI)
- Software media identification number
- Software version, release, and changes status

The frequency of distribution and recipients of the CSR is determined by the Project Management. These reports include the following information.

- Identification of currently approved configuration documentation and configuration identifiers associated with each CI
- CR reports
- Results of configuration audits, status, and disposition of discrepancies
- Traceability of changes from baseline documentation of each CI
- Installation status of deployed releases

The above reports answer basic questions regarding the approved configuration and the implementation status of changes to the baseline.

6.6 Data Recovery and Backup Procedure

6.6.1 Backup

The Change Management Database (CMDB) is used to store information related to the configuration items that will be performed in accordance with current computer facility operations backup process

6.6.2 Recovery

While the decision to recover the CMDB from a backup will certainly result from an emergency situation, it may be part of a location specific or a full recovery restoration HCMIS HE project wide.

In the situation of data loss, the status that is provided with all information necessary to determine a course of action will be notified by CM management. During the restoration, the lost information may require all or only a portion of the backup.

6.7 Release Procedures

6.7.1 Internal Release Process to Test

The steps required to authorize, baseline, build, and release a source code configuration to test for verification during test (Multi-Unit, System Test) will be described by internal release process.

Moreover, the purpose of this process is to provide verification that the current stable source code configuration continues to satisfy all performance and functional requirements (regression testing) and that authorized CRs have been successfully implemented.

6.7.1.1 Build Authorization

The CM management reviews, prioritizes, and authorizes the CRs that will be implemented in the next baseline. Moreover, the CM management requests a formal baseline and builds from the HCMIS HE system developer and emails the version symbol to use and a list of the authorized CRs. The CM management updates the authorized CRs to the "ready for build" state. In addition to this, the release coordinator informs the lead developer to ensure the source code associated with the authorized CRs is properly checked in.

Table 9: Build Authorization Roles and Actions

Step	Role	Action
1	CM management	Schedule Change Configuration Board (CCB) meeting and inform CCB members about location and time
2	CM management	Generate and distribute CR Status Report prior to the CM manager
3	CM manager	Review, prioritize, and authorize which CRs in the resolved state will be implemented into the next build
4	CM management	Email a request to check In code to the lead developer listing authorized CRs
5	CM management	Email a request for formal baseline and build to HCMIS HE System Designer listing authorized CRs and the version identifier
6	RCM management	Update authorized CRs to "ready for build" state.

6.7.1.2 Development Check-In

The lead developer emails a list of the authorized CRs to the development team. Developers that worked on authorized CRs respond back to the lead developer with the name of the workspaces used to work on and fix the authorized CRs. The lead developer (or designated developer) opens each workspace and updates with the current version; builds and verifies.

The lead developer then checks in the associated source code and lists the CR number first in the comments (CR#####). The developer runs Check for locks utility in each workspace to ensure all code associated with authorized CRs are checked in. The lead developer informs the HCMIS HE System Designer when all of the code is checked in.

Table 10: Development Check-In Roles and Actions

Step	Role	Action
1	Lead Developer	Forwards check In request to development team requesting name of workspaces with fixes for authorized CRs
2	Developers	Reply with name of workspaces containing fixes for authorized CRs
3	Lead Developer	Assigns workspace verification to experienced developers

Step	Role	Action
4	Developer	Updates workspace with current version; builds and verifies fix
5	Developer	Checks in modified source code listing CR number first in the comments
6	Developer	Runs Check For Locks utility to ensure all code is checked in
7	Lead Developer	Email HCMIS HE System Designer when all code is checked in

6.7.1.3 Formal Baseline of Source Code

Text query report will be generated by HCMIS HE System Designer from the change management system using the authorized CRs. Moreover, the HCMIS HE System Designer locks the repository and labels the tip revisions of the integration configuration with the Authorized Version identifier.

In addition to this, the HCMIS HE System Designer generates a text baseline compare report from the version control system listing files and revision checks in since the last baseline. HCMIS HE System Designer will contact the lead developer to verify or fix in case of any checked in revisions that do not list an authorized CR number in the comments. The HCMIS HE System Designer combines the CR query with the baseline compare report to produce the baselines release notes.

Table 11: Formal Baseline of Source Code

Step	Role	Action
1	HCMIS HE System Designer	Receives request to build and waits for code check In confirmation
2	HCMIS HE System Designer	Generates a text query report from the change management system using the "ready for build" state
3	HCMIS HE System Designer	Verifies that CR report matches authorized CRs listed in request.
4	HCMIS HE System Designer	Receives code check In confirmation and locks repository
5	HCMIS HE System Designer	Labels the tip revisions with authorized baseline version identifier.
6	HCMIS HE System Designer	Generates a text baseline compare report listing the file and revision differences between the current baseline and the last one

Step	Role	Action
7	HCMIS HE System Designer	Verifies that all new revisions list an authorized CR in its comments
8	HCMIS HE System Designer	Alerts lead developer if revisions missing CR number or have non-authorized CRs in comments and waits for verification or fix
9	HCMIS HE System Designer	Unlocks repository and removes baseline if revisions need to be added or deleted and waits for code check in confirmation – return to step 3
10	HCMIS HE System Designer	Unlocks repository once baseline has been verified

6.7.1.4 Build from Baseline

The HCMIS HE System Designer is a responsible for creating a clean workspace and updates with the source revisions labeled with the Authorized Version identifier. In addition to this task, the HCMIS HE System Designer builds the application from sources providing a non-debug version that deposits the release products in a release directory structure that mimics the installation structure.

The HCMIS HE System Designer contacts the programmer and emails the error messages describing where the build broke in case of the build terminate with errors. If revisions need to be removed or added to the baseline, the current baseline is removed and the baseline process is started over.

Table 12: Build from Baseline

Step	Role	Action
1	HCMIS HE System Designer	Creates new workspace from new baseline with build number in name
2	HCMIS HE System Designer	Builds software for all platforms and monitors for errors
3	HCMIS HE System Designer	Alerts lead developer if build terminates with errors and waits for fix

Step	Role	Action
4	HCMIS HE System Designer	Unlocks repository, removes baseline, baseline compare report, and build view if revisions need to be added or deleted and waits for code check In confirmation before starting baseline process over

6.7.1.5 Package Build Products and Hand-off to Test

When a build is successful, the authorized CRs will be updated to the "ready for test" by HCMIS HE System Designer. Besides, the HCMIS HE System Designer finds the compressed archive file image of the product release area generated by the build and puts it on the local server that can be accessed by the test team.

Table 13: Package Build Products and Hand-off to Test

Step	Role	Action
1	HCMIS HE System Designer	Puts the build generated compressed archive file images of the release products on the local server the test engineer can access
2	HCMIS HE System Designer	Combine CR query with baseline compare report to produce the baseline release notes
3	HCMIS HE System Designer	Emails "build complete" message to project manager, lead developer, and test engineer with the location and name of the build package. Release notes should be attached

6.7.1.6 Verification

From the local server, the Testing Team accesses and copies the compressed archive files to the test machine, and then uncompressed and extracted into the test area.

Table 14: Verification Roles and Actions

Step	Role	Action
1	Tester	Generates a text query report from the change management system using the "ready for test" state
2	Tester	Verifies that CR report matches authorized CRs listed in release notes

Step	Role	Action
3	Tester	Download compressed archive file to the test machine, uncompress and extracts into the test release area as root
4	Tester	Verifies authorized CRs were satisfied and moves them to the closed state
5	Tester	Moves unverified CRs back to opened state
6	Tester	Creates new CRs with submitted state for new defects discovered

6.7.2 External Release Process to Client Environment

The Internal Release Process to Product Test will be followed to produce the build products to be handed off. In addition to the build products, documentation will also need to baselined, verified and released to the deployment team. Negotiation and documentation according the needs of the deployment team will be required for hand-off process. Thus, this process will follow the same procedure as described in the prior section and include additional steps to accommodate the more formal and final packaging of the release to the client environment.

7 ADDITIONAL SECTIONS

7.1 Definition and abbreviation

This document may contain terms, acronyms, and abbreviations that are unfamiliar to the reader. A dictionary of these terms, acronyms, and abbreviations can be found in the next sections.

7.1.1 Abbreviation

- CCB – Change Control Board
- CI – Configuration Item
- CM – Configuration Management
- CMDB – Configuration Management Database
- CML – Configuration Management Library
- CMP – Configuration Management Plan
- CMT – Configuration Management Team
- CR – Change Request
- CSR – Configuration Status Report
- HCMIS HE – Health Commodity Management Information System Hub Edition
- PFSA – Pharmaceutical Fund & Supply Agency
- PM – Project Management
- PTL - Project Team Leader
- QA - Quality Assurance
- SCM – Software Configuration Management

7.1.2 Definitions

- **Baseline**: A Version that has been frozen (Labeled or Archived) in order to achieve Reproducibility, Recoverability, or Trace ability
- **Component**: a file or directory element that can be put under revision control. Components typically include source files, make files, documents, test scripts, design drawings, libraries, binaries, graphics, data files, directories and others.
- **CM Repository** A storage system which contains a copy of all items under Configuration Management, and also contains the information that relates those items to each other and to project deliverables.
- **Configuration:** A set of Components that have been identified as representing a Product.

- **Configuration Items**: Product Configurations are typically broken down into sub configurations like source code, documentation, design artifacts, test artifacts, and so on. These Configuration Items can be managed separately or bundled together and managed as a single unit depending on management style.
- **Current Version:** The Tip Revisions or the last checked in.
- **Revision**: A change to a component. Each revision should be accompanied with a comment describing why the change was required and a brief description of the changes.
- **Version**: A set of component revisions representing a product at any given moment
- **Version Symbol:** A string identifier that revisions are labeled with so they become associated with a baseline. A typical Version Symbol for released configurations use numbers to indicate it's functional and fix level (i.e. 1.0.0). Other unique identifiers like BETA or AUDIT can be used to indicate the reason for the baseline.
- **Integration Configuration:** The Current Version is typically the integration configuration for the development team during Multi-Unit Test. This version is constantly changing as developers check in new changes. A stable Current Version will sometimes be Base lined for Recoverability purposes.
- **Release Configuration**: A Version that has been Base lined in the integration configuration during Product Test and used to produce Products that are handed off to Test to be Verified, QA to be Validated, and/or deployed to the Customer.
- **Multi-Unit Test:** A development life-cycle phase where developers are primarily working on features and are trying to bring an unstable integration configuration to a stable condition.
- **System Test** A development life-cycle phase where a stable integration configuration is under Change Control. Developers primarily work on Change Requests submitted by Test and Enhancements.
- **Maintenance:** A release life-cycle phase where Change Control is still in effect and developers work on Change Requests submitted by Customers with deployed applications.

USAID | DELIVER PROJECT
John Snow, Inc.
1616 Fort Myer Drive, 11th Floor
Arlington, VA22209USA
Phone: 703-528-7474
Fax: 703-528-7480
Email: deliver_project@jsi.com
Internet: deliver.jsi.com