Deerwalk Configuration Management (CM) Process

Version 1.0

[1. Introduction 1](#_Toc356817090)

[2. Configuration Manager 1](#_Toc356817091)

[3. Configuration Management Plan 1](#_Toc356817092)

[4. Configuration Identification 1](#_Toc356817093)

[5. Configuration Control 2](#_Toc356817094)

[5.1. Product Change Control 2](#_Toc356817095)

[5.2. Code Change Control 2](#_Toc356817096)

[5.3. Document Change Control 2](#_Toc356817097)

[6. Configuration Status Accounting 3](#_Toc356817098)

[7. Configuration Audits 3](#_Toc356817099)

[8. Configuration Tools 3](#_Toc356817100)

[9. Tailoring 3](#_Toc356817101)

[10. Appendix A: Deermine Status 4](#_Toc356817102)

# Introduction

This document defines the Configuration Management (CM) process to be followed by Deerwalk projects. The main purpose is to track and control changes in the software built by Deerwalk.

# Configuration Manager

Each project appoints a Configuration Manager who creates and maintains the Configuration Management Plan and ensures that the Configuration Management process is followed in the project.

# Configuration Management Plan

Each project has a Configuration Management Plan which is based on the template available at: <http://www.deerwalk.com/repos/dwfunctions/org/templates/CM_Plan/Deerwalk_Configuration_Management_Plan_Template.xlsx>

# Configuration Identification

Configuration Items (CI) are identified and documented in the Configuration Management Plan.

|  |  |  |
| --- | --- | --- |
| CI Type | Description | Versioning |
| Product | Software product | The recommended format for versioning products is: Major.Minor.Release (Example: 1.0.3) |
| Code | Source code | The Revision Control Tool manages revisions of code. |
| Documents | Product documents | The recommended format for versioning Dynamic documents is: Major.Minor. (Example: 1.3) |
| Others | Anything else that needs to be considered as a CI | Depending on the nature of the CI |

Documents are categorized as per below:

* *Dynamic*: These are documents which need to be versioned. For example: Requirement Specifications, Technical Specifications, Test Plans, etc.
* *Static*: These are documents created at a point in time and need not be versioned or updated . For example: Test Results and Status Reports on a particular date.
* *Active*: These are documents which undergo frequent changes. For example: Schedule.

# Configuration Control

Any change to baselined Configuration Items follows the following Configuration Control process.

## Product Change Control

* All change requests (features/enhancements/defects) of the software product are logged in the Request/Defect Tracking Tool. [For the flow in Deermine, see .]
* Impact analysis is done to assess the effects of the change request.
* Work starts on the change request only after approval of the concerned stakeholders. [This typically involves the Account Manager (AM), US Development Manager (USDM), Project Manager (PM), Development Manager (DM), and the Quality Assurance Manager (QM)]
* Once the change request is approved, it is implemented and pushed to QA/UAT Environment.
* Once the testing passes, the change requests are implemented in the Production Environment.
* Release Notes, listing the addressed change requests, is distributed.

## Code Change Control

* Revision Control Tool is used to manage code and track revisions.
* Code commits are made only against an approved change request. (See ]
* Main development is performed in a defined location in the repository (*Trunk*).
* *Branches* are created to try out new features or to fix something without impacting the main line of development. Stable code in a Branch is then merged with Trunk.
* *Tags* are created to maintain static snapshots of the product at a particular stage (say, a release). No changes are made to Tags.
* Production releases are always tagged.

## Document Change Control

* Changes to dynamic documents like Requirement Specifications, Technical Specifications, Test Plans, User Guides, etc, are made by respective owners after being reviewed by concerned stakeholders.
* Revision Control Tool may be used to store documents.

# Configuration Status Accounting

The Configuration Manager ensures that all configuration items are recorded, kept current, and their status reported when necessary. The Revision Control Tool and the Request/Defect Tracking Tool reflect an up-to-date status of code and request/defect respectively.

# Configuration Audits

Configuration Audits are performed periodically:

* *Internal Configuration Audit:* To be performed by the Configuration Manager of the project.
* *External Configuration Audit:* To be performed by an independent Audit Team outside of the project.

# Configuration Tools

The organization provides the following configuration tools but each project may choose their own tool depending on the nature of the project and the preference of the client.

* Revision Control Tool:
  + *Git*: Recommended
  + *Subversion*: This will be phased out once all current projects migrate to Git
* Request/Defect Tracking Tool:
  + *Deermine*: Based on Redmine

# Tailoring

Any tailoring to the defined configuration management process documented here must be agreed upon in advance by the concerned stakeholders of the project.

# Appendix A: Deermine Status

|  |  |
| --- | --- |
| Status | Description |
| New | The change request is new and needs impact analysis and approval. Once approved, it is assigned to the concerned Developer for resolution. |
| Feedback | The change request requires more information. |
| In Progress | The change request is being addressed. |
| Resolved | The change request is implemented and ready for testing. |
| Verified | The changes request is tested in QA/UAT Environment. |
| Deferred | The change request is deferred for a later release. |
| Closed | The change request is implemented in Production Environment |
| Rejected | The change request is rejected. |