**

Track Model

test Plan

Version 1.0

Ryan Mohan

March 22, 2012

|  |  |  |
| --- | --- | --- |
| List Of Revisions | | |
| Date | Name | Description |
| 3/22/2012 | Ryan Mohan | Initial design document v 1.0 |

# Table of Contents

[List Of Revisions 2](#_Toc320188829)

[Table of Contents 2](#_Toc320188830)

[1 Introduction 3](#_Toc320188831)

[1.1 Purpose 3](#_Toc320188832)

[1.2 Scope 3](#_Toc320188833)

[1.3 Reference 3](#_Toc320188834)

[1.4 Definitions and Abbreviations 3](#_Toc320188835)

[2 Test Items 3](#_Toc320188836)

[3 Features to be Tested 3](#_Toc320188837)

[4 Testing Strategy 4](#_Toc320188838)

[4.1 Unit Testing 5](#_Toc320188839)

[4.2 Item Pass/Fail Criteria 5](#_Toc320188840)

[4.3 Test Deliverables 5](#_Toc320188841)

[4.4 Responsibilities 5](#_Toc320188842)

# Introduction

## Purpose

The purpose of this document is to define the testing plan for the Track Model and its associated classes.

## Scope

This document describes the testing approach and specific scenarios that will be tested in Track Model

## Reference

1. IEEE-830 Software Requirements Specification
2. NSC-009 PAAC Bid Package
3. Centralized Traffic Control Graphical User Interface Software Requirements Specification
4. Bazinga System SRS
5. Bazinga-Mohan-Ryan-SubSRS
   1. See Section 3

## Definitions and Abbreviations

**Block** – a section of a railway line

**CTC** – Centralized Traffic Control

**GUI** – graphical user interface

**Region** – the track between two switches

**Signal** – The wayside signal next to each track block

# Test Items

This document covers only the testing plan for the Track Model independent of other modules in the system, with the exception of data contracts from the other modules needed for testing purposes. The test plan is intended for deliverable software candidates for the final product release. The inputs received from other system modules will be simulated during Track Model only testing, but will be tested for real system interaction during the full system tests.

# Features to be Tested

The test plan will target the following features of the Track Model

* The interface between the CTC and office GUI (UI)
* The interface between the track and track controller
* The ability for a dispatcher to:
  + Open Block
  + Close Block
  + Edit Block Size
  + Open Station
  + Close Station
  + Properly relay signal state to CTC
  + Relay track condition, grade, and elevation
  + Relay switch states to the CTC and Track Controller
* The ability to input track layouts with:
  + Different block sizes
  + Different block sizes
  + Different Grades and Elevations
  + Input Switches, Stations, Tunnels, and RR Crossings
* The ability for every block to reach the following 3 failure modes
  + Broken Rail
  + Track Circuit Failure
  + Power Failure
* Each Block shall be able to properly take the following inputs from the Track Controller
  + Signal State
  + Speed Limit
  + Acceleration Limit
  + Deceleration Limit
  + Authority
* The ability for each block to issue the following outputs to the Track Controller
  + Speed Limit
  + Authority
  + Block State
    - Open
    - Closed
    - Failure Modes (see above)
  + RR Crossing State
  + Signal State
* The ability for blocks of any type to be added to a Region of any size

# Testing Strategy

The testing for the Track Model shall be performed through Unit Testing. Unit testing will test all of the above mentioned inputs and outputs in all possible conditions (valid input, invalid input, null) for each input and track type. The Unit Testing shall be performed on the Track Model individually and also ran through the system as a whole prior to release.

## Unit Testing

The CTC GUI software will be unit tested using Microsoft Visual Studio 2010. The Track model software author will write corresponding unit tests to cover, at a minimum, each public method of every class in the software package. Each public method will be tested with various inputs to cover feasible scenarios (e.g. good data, invalid data, null data). Similar test cases will also be written to cover functional paths, i.e. methods that call other methods. The unit tests will be compiled into a program which can be executed with Visual Studio 2010. The results will be recorded in a deliverable test report.

## Item Pass/Fail Criteria

All unit tests and module tests are expected to pass. In the event that one or more tests fail, an assessment will be made by the development team to determine the severity of the error(s). The team will then decide on a course of action to resolve or mitigate the error(s), or to release the software with the defect(s) present. Each error will be assigned a severity of High, Medium, and Low depending on the risk to the system or the risk to non-fail-safe actions. The possible errors will then be handled and fixed in priority of their severity.

## Test Deliverables

The following documents are to be delivered with the final software release:

* Unit Test Output and Report for the Track Model
* Unit Tests Outputs for the System as a whole

## Responsibilities

The software author is responsible for writing the unit test cases for each testable component of the software. The author is also responsible for writing the module test cases document. A developer other than the author is responsible for verifying the completeness and correctness of the unit tests and module test document. The same developer or another developer is responsible for executing the unit test program and module test and composing a test report for each. The software development team is collectively responsible for deciding on actions to be taken in the event that one or more test cases fail. The software author is then responsible for taking the decided action to resolve or mitigate the errors.