**

Centralized Traffic Control Graphical User Interface test Plan

Version 1.0

March 22, 2012

PAAC Demonstration System

|  |  |  |
| --- | --- | --- |
| List Of Revisions | | |
| Date | Name | Description |
| 3/22/2012 | Jeremy Nelson | Initial design document |

# Table of Contents

List Of Revisions 2

Table of Contents 2

1 Introduction 3

1.1 Purpose 3

1.2 Scope 3

1.3 Reference 3

1.4 Definitions and Abbreviations 3

2 Test Items 3

3 Features to be Tested 3

4 Testing Strategy 4

4.1 Unit Testing 4

4.2 Module Testing 4

4.3 Item Pass/Fail Criteria 5

4.4 Test Deliverables 5

4.5 Responsibilities 5

# Introduction

## Purpose

The purpose of this document is to define the testing plan for the Centralized Traffic Control (CTC) Graphical User Interface (GUI) module of the PAAC transit control prototype system.

## Scope

This document describes the testing approach and specific scenarios that will be tested in the CTC GUI module.

## Reference

1. IEEE-830 Software Requirements Specification
2. NSC-009 PAAC Bid Package
3. Centralized Traffic Control Graphical User Interface Software Requirements Specification

## Definitions and Abbreviations

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

**CTC** – Centralized Traffic Control

**GUI** – graphical user interface

# Test Items

This document covers only the testing plan for the CTC GUI 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 and does not necessarily apply to interim software versions.

# Features to be Tested

The test plan will target the following key features of the CTC GUI:

* User login to the transit system
* Display of the transit system state in a track layout view
* Display of information specific to a track component
* Display of information specific to a train in the system
* Ability to send suggestions, such as speed limit and authority, to track components
* Display of track block information in a table format
* Display of train information in a table format
* Display of scheduling information in a table format
* Ability to schedule trains and send routing information accordingly

# Testing Strategy

The testing for the CTC GUI will be split into two primary categories: Unit testing and Module Testing. The test cases should be reviewed for completeness prior to execution by a software developer other than the software author. The tests should also be run and composed into a report by a software developer other than the software author.

## Unit Testing

The CTC GUI software will be unit tested using Microsoft Visual Studio 2010. The CTC GUI 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.

## Module Testing

Test cases will be written to exercise the possible user interactions with the CTC GUI. The test cases will be targeted at verifying expected GUI responses and the messages sent to the transit system. Since the testing is intended to only test the CTC GUI module, transit system interfaces will be replaced with stubbed classes to mimic system responses. The stubbed classes will record the messages passed to and from it in a log file. The log file will be compared with the expected output to verify the CTC GUI calls the interfaces correctly. Primary scenarios to be tested include the following:

* Loading a track layout and verifying the correct track display on the GUI
* Selecting various track components and verifying the correct information display and command options
* Suggesting a valid speed limit to a track block and verifying the correct set speed limit command is sent to the correct track controller
* Entering an invalid speed limit and verifying the GUI rejects the request
* Suggesting a valid authority to a track block and verifying the correct set authority command is sent to the correct track controller
* Entering an invalid authority and verifying the GUI rejects the request
* Selecting a train and verifying the correct information display and command options
* Suggesting a train route and verifying the correct set route command is sent to the correct track controller(s)
* Verifying the correct display of track block information in the table view screen
* Entering speed limit and authority suggestions from the table view screen
* Verifying the correct display of train information in the table view format
* Verifying the correct display of scheduling information in the table view format
* Setting train scheduling information and verifying the correct scheduling information is sent to the correct train(s)
* Dynamic testing with a simple simulator that changes the track state in time to verify correct updating of the track display
* Robustness testing with a simple simulator to verify the GUI is responsive after several hours of run time

## 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.

## Test Deliverables

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

* Unit Test Output and Report
* Module Test Cases
* Module Test Report

## 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.