Table of Contents

[1 General 2](#_Toc491690998)

[1.1 Purpose 2](#_Toc491690999)

[1.2 Scope 2](#_Toc491691000)

[2 Overview 3](#_Toc491691001)

[3 Architecture 4](#_Toc491691002)

[3.1 Components 4](#_Toc491691003)

[3.1.1 TestStationStatus 4](#_Toc491691004)

[3.1.2 TestStationStatusDomain 4](#_Toc491691005)

[3.1.3 TestStationStatusInfrastructure 4](#_Toc491691006)

[3.1.4 UnitTests 4](#_Toc491691007)

[3.2 Application start up 4](#_Toc491691008)

[3.3 Test station communication 4](#_Toc491691009)

[3.4 Next steps 5](#_Toc491691010)

[3.4.1 Add a Rest API for communication with test stations / a dynamic view of state. 5](#_Toc491691011)

[3.4.2 Any other features driven by user feedback 5](#_Toc491691012)

[3.4.3 Use the built in account management to feed email addresses 5](#_Toc491691013)

[3.4.4 Reporting 5](#_Toc491691014)

[4 Adding new test stations 6](#_Toc491691015)

[5 Deployment 8](#_Toc491691016)

[6 3rd part components 9](#_Toc491691017)

[6.1 Autofac 9](#_Toc491691018)

[6.2 Datatables.aspnet.-MVC 9](#_Toc491691019)

[6.3 EntityFramework.SqlServerCompact 9](#_Toc491691020)

[6.4 Microsoft.AspNet.SignalR 9](#_Toc491691021)

[6.5 Moq 9](#_Toc491691022)

[6.6 Newtonsoft.Json 9](#_Toc491691023)

[6.7 Nlog 9](#_Toc491691024)

[6.8 PdfMake 9](#_Toc491691025)

# General

## Purpose

This document outlines the software implementation of a web application used to control ATS test station.

## Scope

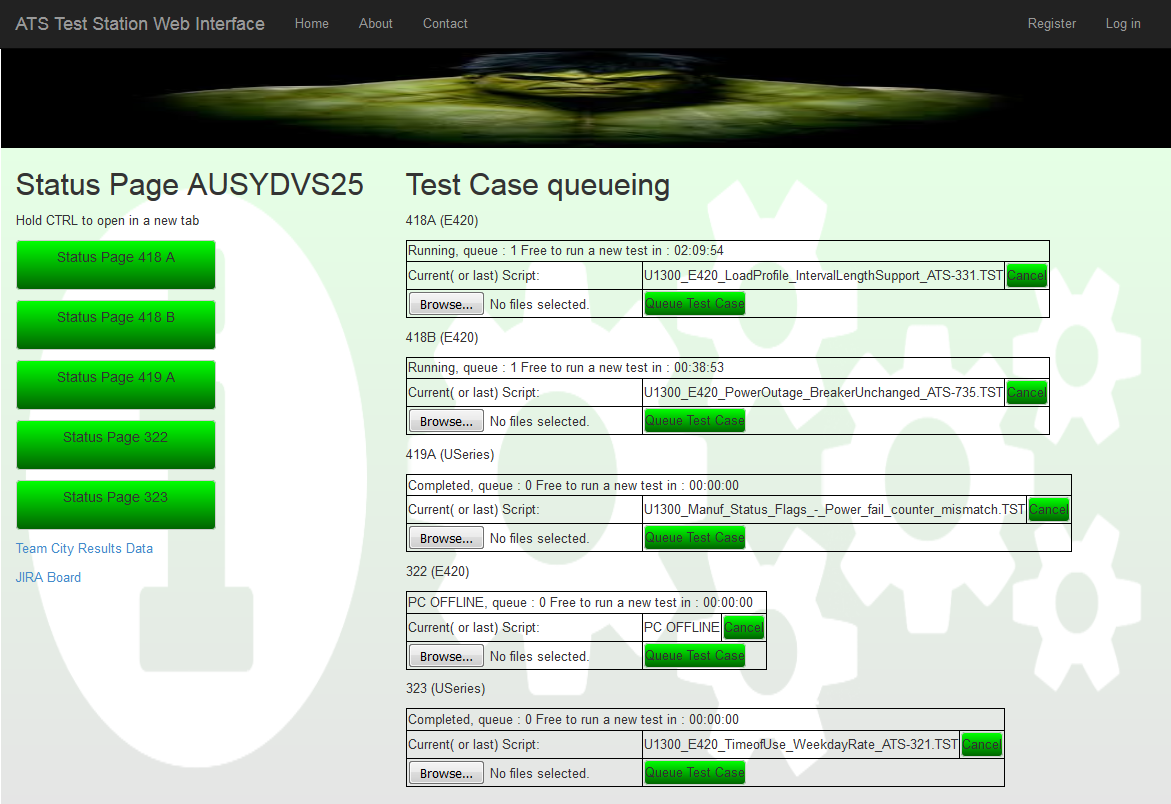
This document pertains only to the software implementation.

# Overview

The webpage is a MVC application that provides a simple front end for each test station. It is designed for the test case developer to quickly see what station to they should run the test on and to provide an overview of what is in progress.

This same interface can be used by any interested party to run tests, for example a firmware engineer could place a new version on a test station (currently a manual process) and then select any test case from the ‘Tested’ folder to create a test plan to cover the nature of code change.

This application was created for the initial need for an overview of the state of the test systems, it has been centralised and improved based on feedback from users.



# Architecture

## Components

The solution is made up of the following components –

* TestStationStatus
* TestStationStatusDomain
* TestStationStatusInfrastructure
* UnitTests

### TestStationStatus

This is the main application. It contains the logic for displaying the pages, references the other libraries.

### TestStationStatusDomain

This module contains all the domain level objects.

### TestStationStatusInfrastructure

This module contains all the services and core functionality.

### UnitTests

Test project to check features provided by the infrastructure library.

## Application start up

When the application is first viewed after a software update the web page will check to see if the background service is running, if not it will start this service and wait for the model to become ready. The model is created on setup and then updated every 10 seconds. The model reflects the current state of the test stations.

## Test station communication

Currently the data is polled every 10 seconds, a decision was made to work this way due to fact that making modifications to the application that runs the test scripts requires a long and difficult approval process. Ideally the communication would move to a rest api on the server, with each test station registering itself on the server.

Each station is checked using network shares. The test station updates a few files that show the current state of the application. Other state is determined simple by the number of files located within certain folders.

## Next steps

### Add a Rest API for communication with test stations / a dynamic view of state.

Create an automatic registration and update process, so new stations will automatically appear on the Web interface without code changes. A deregistration or archive process for the decommissioning of test stations.

### Any other features driven by user feedback

The users of the system have driven a lot of the current feature set. This systems goal is to make the writing of test cases more efficient. Addition users could be identified and their requirements accessed.

### Use the built in account management to feed email addresses

Once a test script has been completed there is currently an email send to the person who triggered the test. The current application uses some hard coded data and PC network names to resolve users, this should be replaced with the built in ASP.net user system.

### Reporting

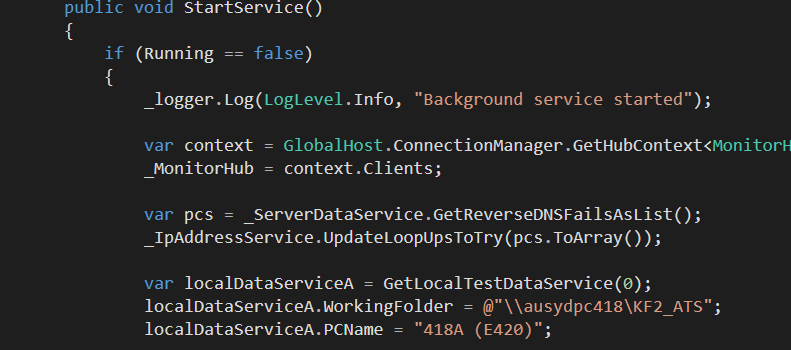
Some TCD like reporting should be considered.

# Adding new test stations

The configuration of the available test stations is currently hard coded, this means that when a new test station is added these hard coded values must be changed or added to.

In ‘RefreshClientService’ class contains the network path to each of the test stations.

The home page shows information for each of the test stations, in order to add a new stations this page must be modified to display the setup for the new station. This page has some dynamic design so adding a new station is mostly a cut-n-paste exercise while changing some id fields.



Test station

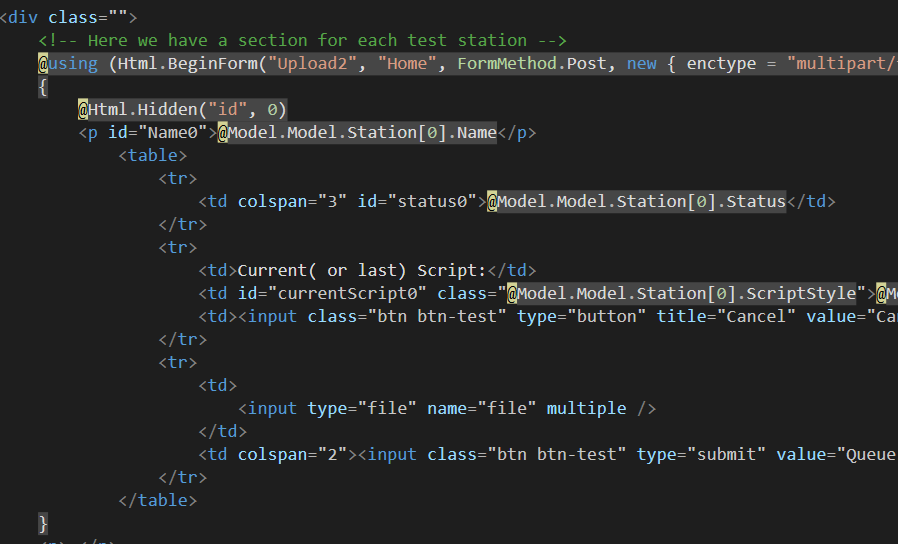
Example new test station details –

*var localDataServiceD = GetLocalTestDataService(<next free id>);*

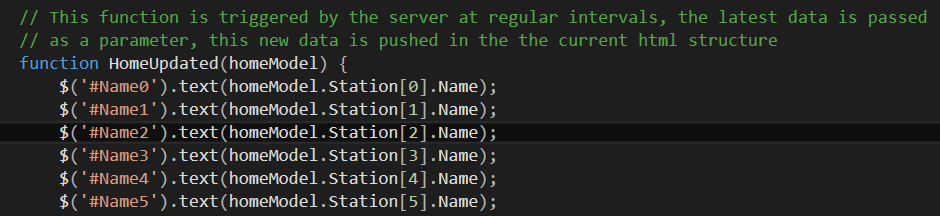
*localDataServiceD.WorkingFolder = @"<Network path to KF2\_ATS share>";*

*localDataServiceD.PCName = "<Name of test station>";*

HTML block for a test station –



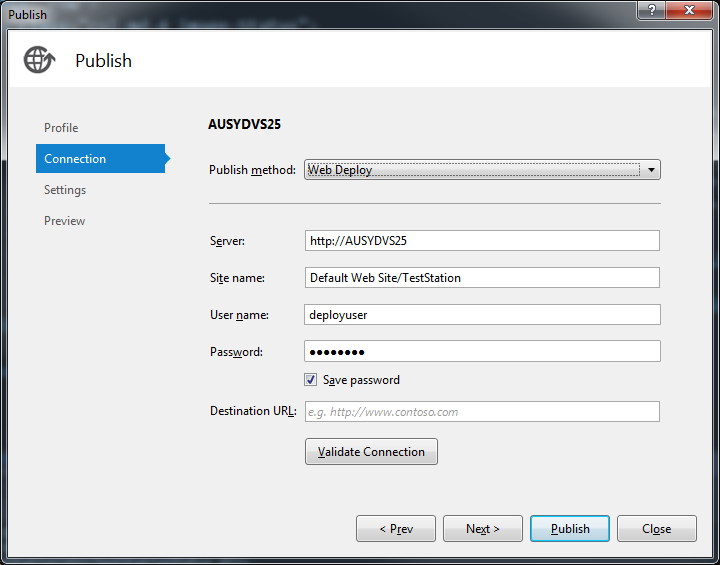
Server triggered model update (signalR)



# Deployment

The web application has not special deployment requirements and the publish wizard can be used. There is a database and this will be created if not already present. This application is currently published to <http://ausydvs25/TestStation/> (internal server).

The publish details have been saved as part of the project.



# 3rd part components

There are lots of components installed as standard when setting up a MVC project. This list contains the ones which have been added.

## Autofac

Provides IOC features.

## Datatables.aspnet.-MVC

This is used to show JQuery data tables.

## EntityFramework.SqlServerCompact

This is used to provide a database connection.

## Microsoft.AspNet.SignalR

This is used to provide client update notifications.

## Moq

This is used to create test objects when writing unit tests.

## Newtonsoft.Json

This is the most common library used to create and receive json data.

## Nlog

This is a generic logging library.

## PdfMake

This is used to create pdf files from results set on the client side.