Blackhawk Checking Application

1 Introduction

This document gives an introduction of designing the Blackhawk Checking Application. Blackhawk checking application is a software program that can test new produced Blackhawk units in the factory. It can also update the passed Blackhawks to the Blackhawk database and generating a testing report.

1.1 Business Description

Before the Blackhawk units enter the marketplace, the factory has to check whether the products work properly. This is being done manually at this stage. We need an application that can assist the production staff to do this job more efficiently.

1.2 Glossary

shall
: Used in the Requirements section 'shall' means that the item is absolutely necessary as stated. Example: *The product shall support at least one million master records.*

should
: Used in the Requirements section 'should' means the item is desirable, but not required. Wishy-washy, but often unavoidable. Example: *The product should maintain the same order of data entry fields as current system XYZ.* These requirements will be implemented where feasible within the other constraints.

TBD
: To Be Decided/Determined.

tester
: Person(s) who will use the completed product of this specification.

Unit
: The Blackhawk unit

Unit report
: The report sent to the Blackhawk server by the Blackhawk unit.

BH
: Blackhawk

2 Goals

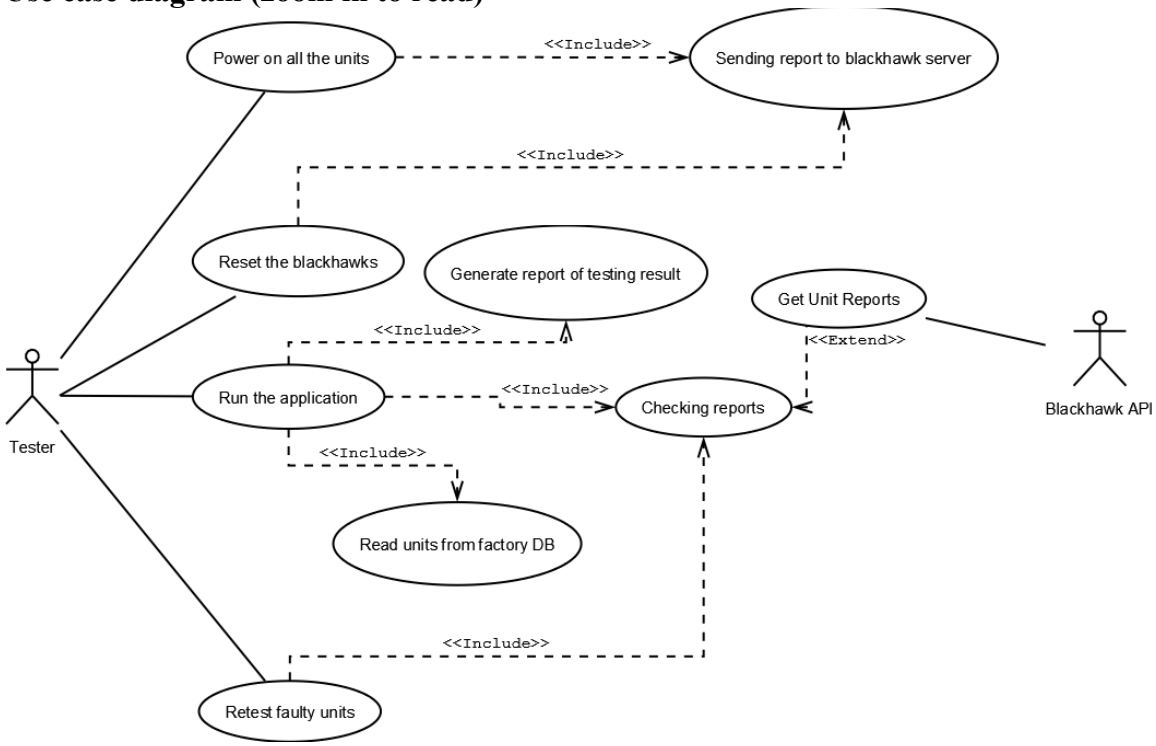
The goal of this application is increasing the efficiency of checking new produced Blackhawk units.

3 Requirements

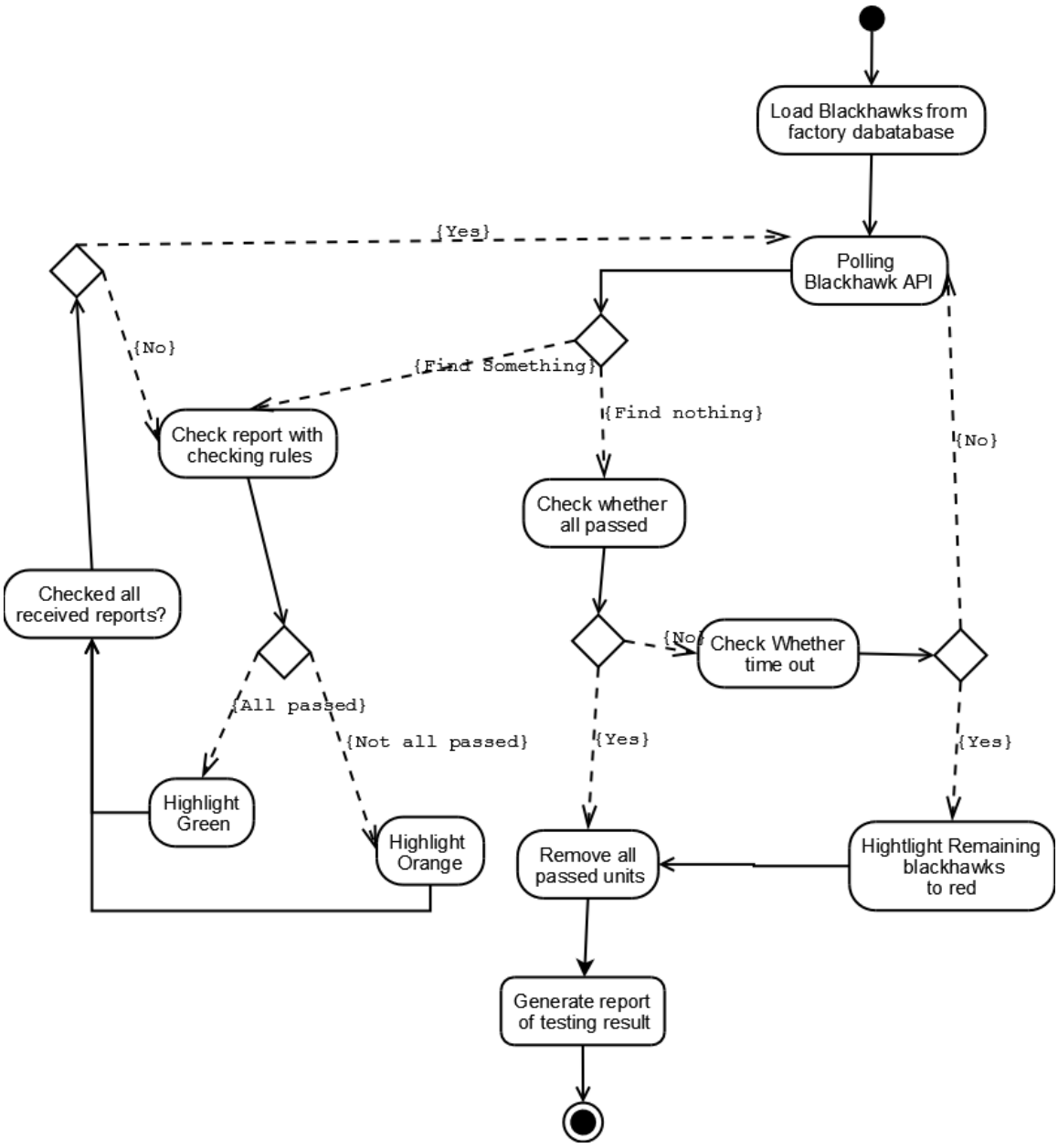
- The application shall have a login panel that allows the user to login.
- The application shall have the ability to read and update the factory database.
- The application shall have the ability to read the received units reports from the Blackhawk database.
- The application shall highlight the pass units with green color.
- The application shall highlight the faulty units with orange color.
- The application shall highlight the units whose reports cannot be detected with red color.
- The application shall be able to remove the pass units from the display window.
- The application shall be able to retest the units that are highlighted red.
- The application shall be able to add the new pass units to the Blackhawk database.
- The application shall be able to generate a report of the testing result.

Getting started

Use case diagram (zoom in to read)



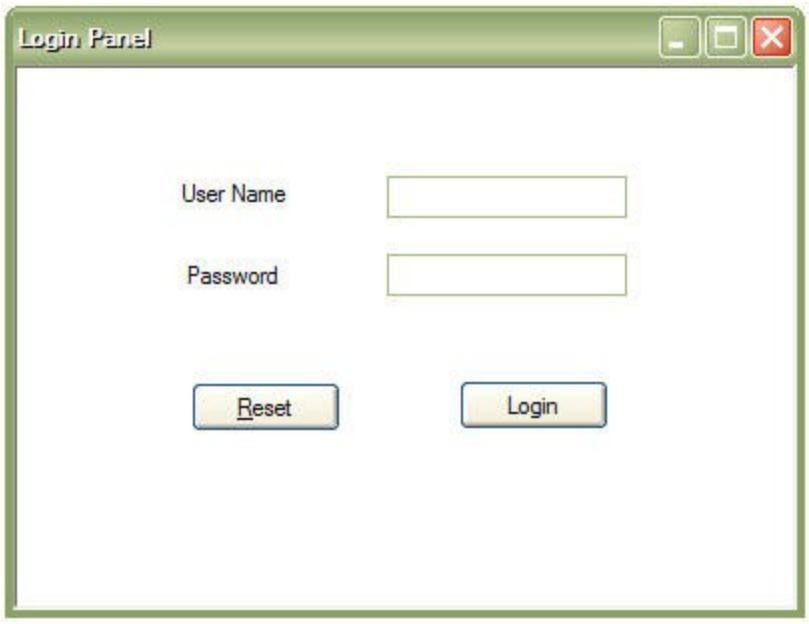
State machine diagram of the checking process of the application (zoom in to read)



Note: The interfaces in this specification are used as a guide. The application doesn't need to follow these interfaces exactly.

Login Panel

The Login Panel



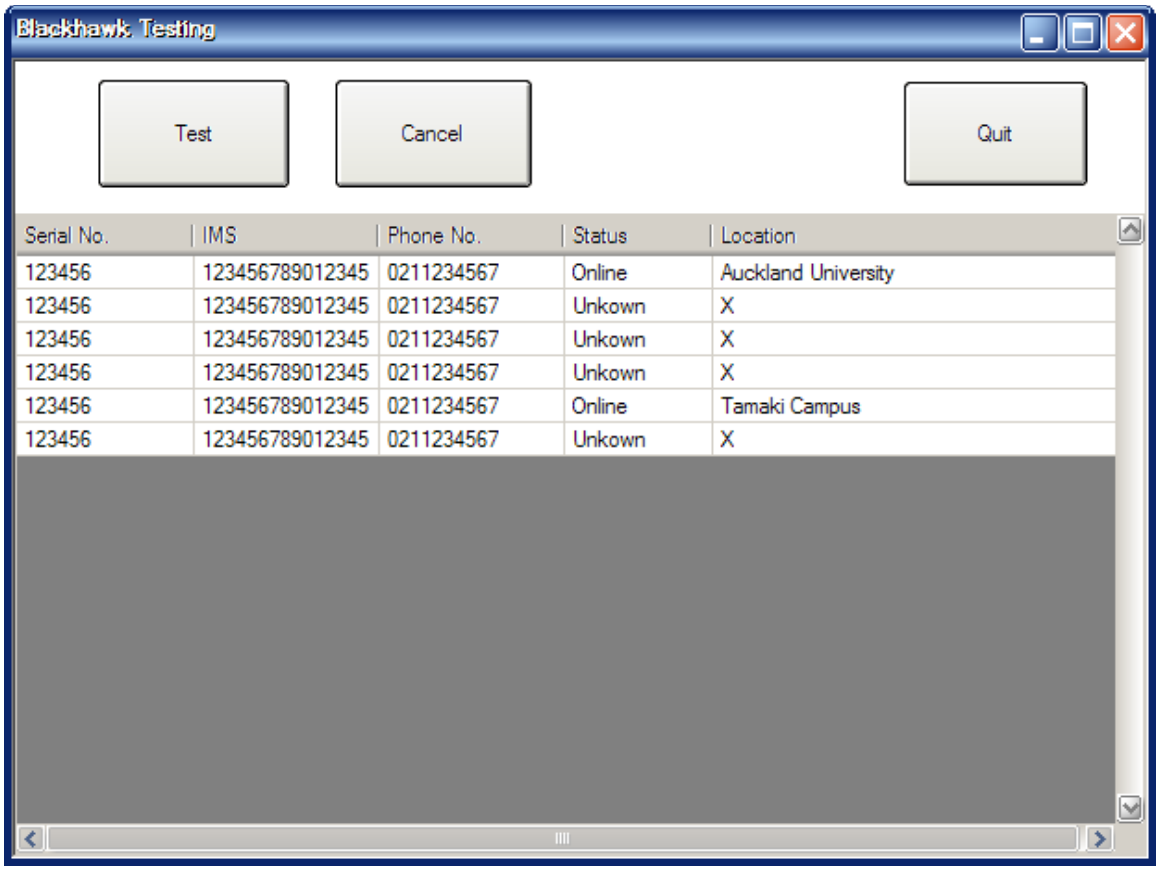
The login panel allows the tester to login.

If login failed, the application shall give an error message.

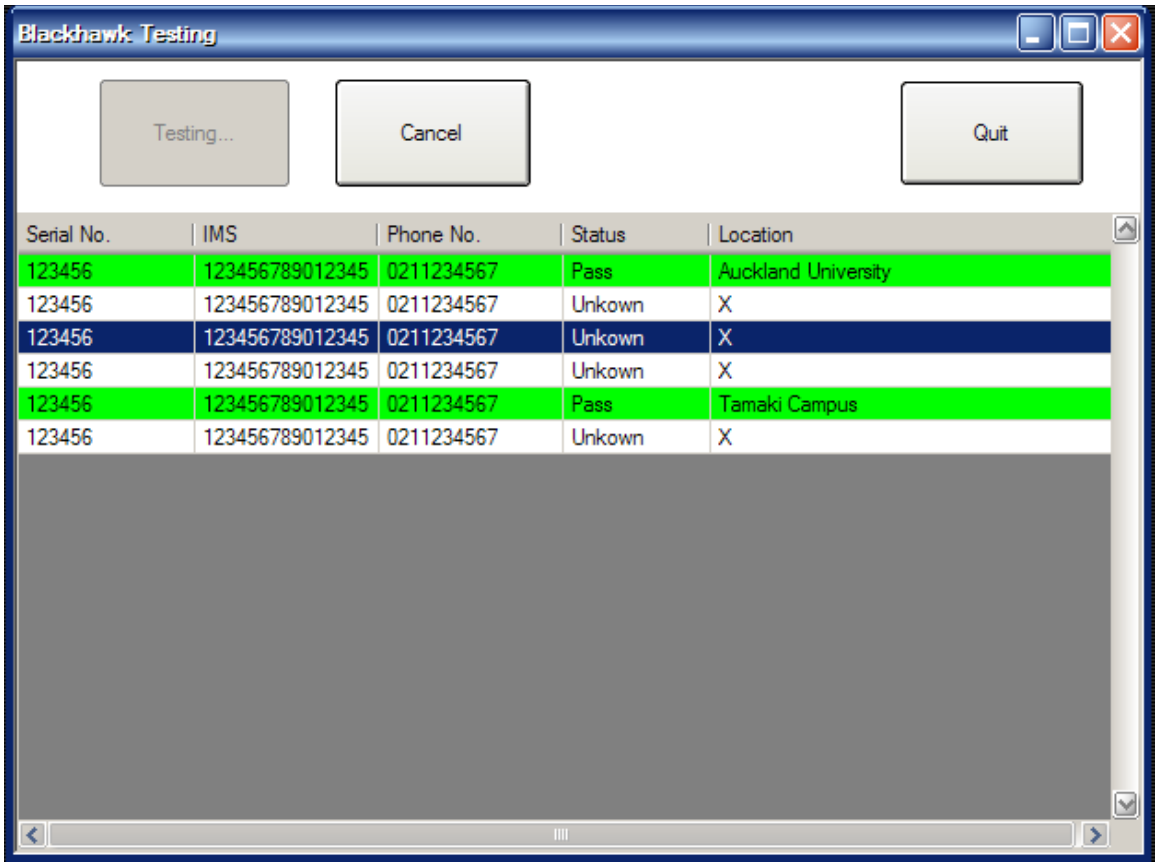
The Main Program

Notes: The status “unkown” is mistake spelling of “unknown”

After Login, The application will read the units from the factory database automatically. All the units shall be displayed in the application window.


Blackhawk Testing
Test
Cancel
Quit
Serial No.
IMS
Phone No.
Status
Location
123456 123456789012345 0211234567 Online Auckland University
123456 123456789012345 0211234567 Unkown X
123456 123456789012345 0211234567 Unkown X
123456 123456789012345 0211234567 Unkown X
123456 123456789012345 0211234567 Online Tamaki Campus
123456 123456789012345 0211234567 Unkown X

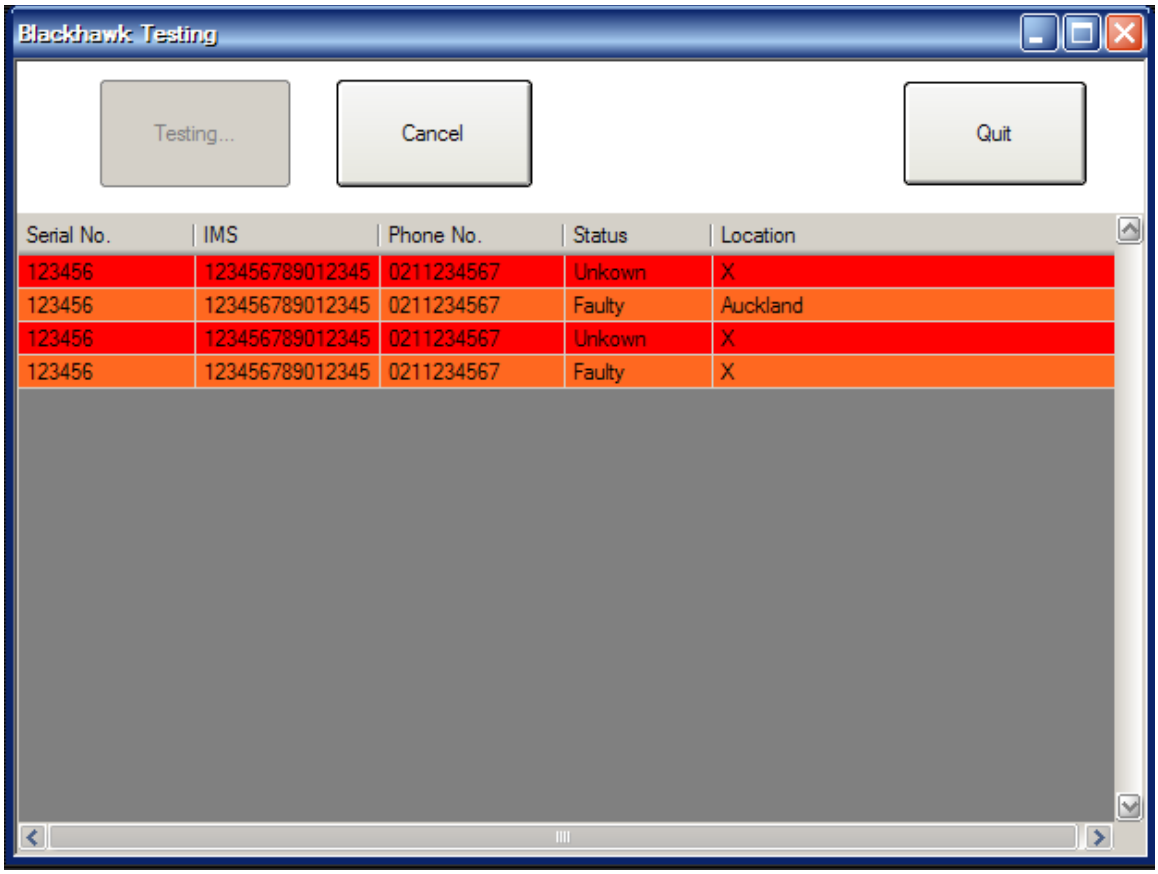
The test button shall be disabled after clicking. Then the application will start to get the unit reports by polling Blackhawk API.



The pass units shall be highlighted green.
The faulty units shall be highlighted orange.

The program will stop testing if all units pass or time out.

The pass units will be removed from the display panel. The units whose report cannot be got will be highlighted red.


Blackhawk Testing
Testing...
Cancel
Quit
Serial No.
IMS
Phone No.
Status
Location
123456
123456789012345
0211234567
Unkown
X
123456
123456789012345
0211234567
Faulty
Auckland
123456
123456789012345
0211234567
Unkown
X
123456
123456789012345
0211234567
Faulty
X

Report
A report with the date as the part of file name will be generated in the reports folder.

The report should be a XML file that contains all the testing results. **The report format**

```
<!--report_1-1-2008.xml-->
<Testing Results>
        <Unit>
                <Number>1</Number>
                <Serial No>123456</Serial No>
                <IMS>123456789012345</IMS>
                <Phone No>0211234567</Phone No>
                <Status>Unknown</Status>
                <Location>X</Location>
        </Unit>

        <Unit>
                <Number>2</Number>
                <Serial No>123456</Serial No>
                <IMS>123456789012345</IMS>
                <Phone No>0211234567</Phone No>
                <Status>Faulty</Status>
                <Faulty_Location>Sydney</Faulty_Location>
                <Faulty_IMS>987654321012345</Faulty_IMS>
        </Unit>
        <Unit>
                <Number>3</Number>
                <Serial No>123456</Serial No>
                <IMS>123456789012345</IMS>
                <Phone No>0211234567</Phone No>
                <Status>Pass</Status>
                <Location>Auckland University</Location>
        </Unit>
        <Unit>
                <Number>4</Number>
                <Serial No>123456</Serial No>
                <IMS>123456789012345</IMS>
                <Phone No>0211234567</Phone No>
                <Status>Pass</Status>
                <Location>Auckland University</Location>
        </Unit>
        <Unit>
                <Number>5</Number>
                <Serial No>123456</Serial No>
                <IMS>123456789012345</IMS>
                <Phone No>0211234567</Phone No>
                <Status>Unknown</Status>
                <Location>X</Location>
        </Unit>
</Testing Results>
```

The faulty items will be recorded as an element with a name start with “Faulty_”

The xml file can be converted to more human-readable format if need.

For example, an html file likes this

Serial No	IMS	Phone No	Status	Location
123456	123456789012345	0211234567	Pass	Auckland
123456	123456789012345	0211234567	Unkown	X
123456	123456789012345	0211234567	Faulty	Sydney
--Faulty Location: Sydney				
--Faulty IMS: 987654321012345				
123456	123456789012345	0211234567	Unkown	X
123456	123456789012345	0211234567	Faulty	X
--Faulty IMS: 987654321012345				

The configuration file

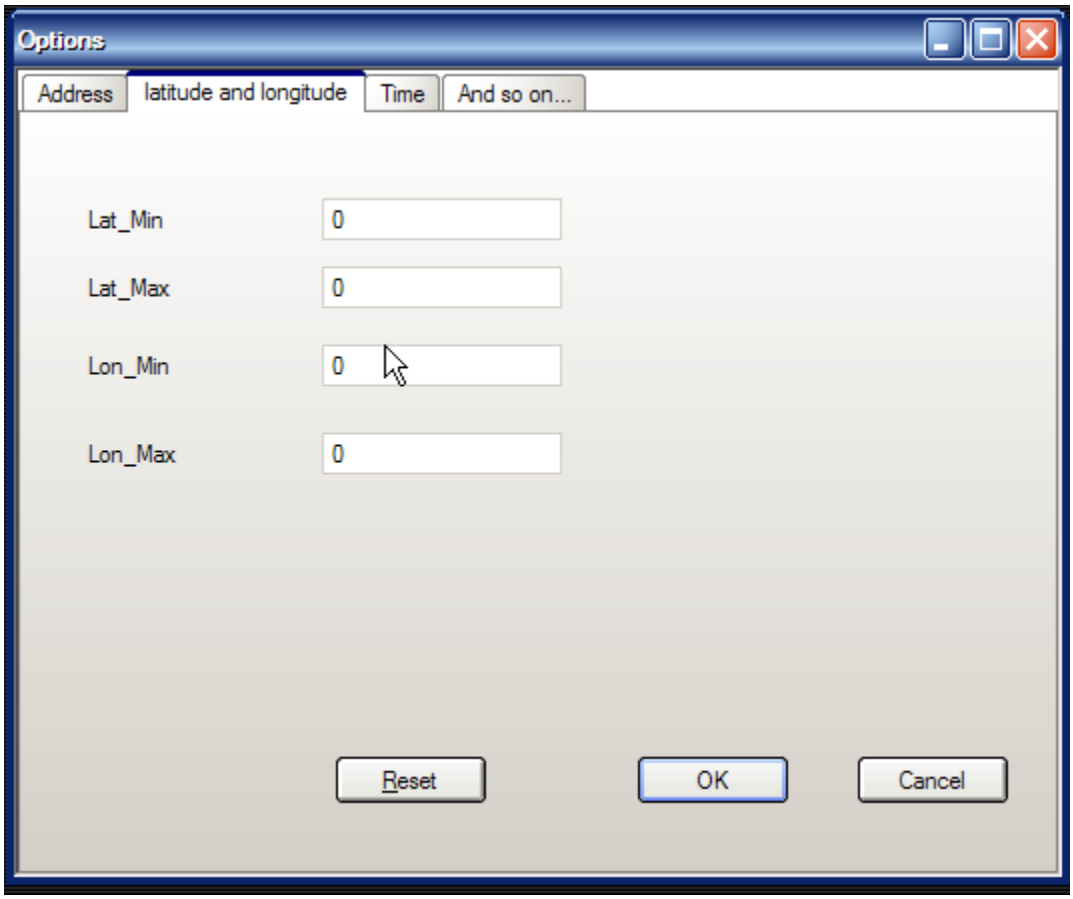
The configuration file should be an xml file like this:

```
<!—file name: buta.exe.config-->
<?xml version="1.0" encoding="utf-8"?>
<configuration>

        <appSettings>
                <add key="TimeOut" value="15" />
                <add key="FactoryAddress" value="Auckland University" />
                <add key= "Interval" value="2" />
                <add key= "Lat_Min" value="0" />
                <add key= "Lon_Min" value="0" />
                …
        </appSettings>
</configuration>
```

The tester can configure the configuration by editing this file directly. The application may supply an option panel to allow the tester to edit the configuration using the application.

For example:



If the attribute is not in the configuration file, use the default value.

These are the attributes that are configurable

TimeOut

Application will stop pulling BH API after this amount of time.
Default Value: 30 minutes.

Interval

Application will wait this amount of time for next pulling.
Default Value: 2 minutes.

Lat_Min

The minimize value that can be accepted as a latitude.
Default: TBD

Lat_MAX

The maximize value that can be accepted as a latitude.
Default: TBD

Lon_Min

The minimize value that can be accepted as a latitude.
Default: TBD

Lon_Max

The maximize value that can be accepted as a latitude.
Default: TBD

Address

The factory address
Default: Auckland University

The Blackhawk API

TBD