TEAMS-Mobile: BlackBerry

Sponsored by Qualtech Systems, Inc.

Advisor: Dr. John Chandy, ECE
Advisor: Kristian Balinski, QSI

C.J. Manville, EE & CSE
Rockwell Schrock, CompE
Steve Swirsky, EE

Overview

TEAMS-Mobile is an extension of the Qualtech Systems, Inc. (QSI) diagnostic software service that runs on mobile computing devices. The TEAMS-Mobile: BlackBerry project is writing a native application to implement all of the necessary features to conveniently run the automated diagnostic software on a BlackBerry device.

Diagnostics are important in every industry because inevitably all machinery over a long enough time span will break down and need diagnostic attention. Most diagnostic procedures are done as individual case-based problems, where a technician tries to use a brute-force method to figure out that particular failure. This method is time consuming and inefficient on a large scale because it takes many hours of highly-trained technicians.

There is an alternative to case-based diagnostic analysis, and that is model-based analysis. Model-based analysis offers many advantages over case-based, including: more efficient diagnosis, lower necessary technician expertise, more efficient preparation for troubleshooting, and a dynamically evolving and improving diagnosis system that improves from each case. QSI uses this model-based approach, providing software to model how a particular system fails. To create the diagnosis model, an expert of the given system is required to go through the possible outcomes in the beginning of the process. This can be considered by some to be a large initial time and effort investment, but it allows for much greater troubleshooting efficiency later on, utilizing the expertise of someone with intimate knowledge of the system.

Background

The TEAMS-RDS server is the brains of the TEAMS suite of applications. It is a powerful model analysis tool that has a convenient Web interface for running diagnostics. It is one of QSI's flagship products. However, its Web interface must be accessed through a full-blown Web browser client application. This requires both a computer capable of running a browser and an active, uninterrupted network connection.

The interface for TEAMS-RDS is designed for large laptop displays. There is a mobile Web interface, but it is crude and out-of-date. It does not support the new functionality of modern mobile device browsers. Again, this still requires an active network connection and a decently fast browser.

Last year's Qualtech senior design team began tackling these issues by developing a mobile application called TEAMS-Mobile. This application runs on the Windows Mobile platform. It runs through a troubleshooting session based solely on a static XML decision tree that has been generated by TEAMS-RDS. Therefore, it always functions in a stand-alone, "offline" mode without ever talking directly to TEAMS-RDS.

Solution

To overcome the primary limitations of a Web application, we are instead utilizing the native application programming interface (API) of the BlackBerry mobile device. This includes the user interface generation as well as parsing and traversing of the XML decision tree provided by the server. Initially we explored using Web Services to communicate with the server. This would be expensive for Qualtech employees to develop, and is not possible to complete in our short development window.

The existing TEAMATE application, which has similar functionality to TEAMS-Mobile, is used on a notebook computer. Many clients have systems stored in sensitive areas where an outside laptop cannot be brought in for security purposes. Therefore, a technician would not be able to bring in a notebook to use TEAMATE to troubleshoot a system there. However, many of these places allow smart phones where laptops can't go. This is where TEAMS-Mobile will come into play.

We are also working to expand TEAMS-Mobile into other mobile platforms. Windows Mobile was supported with last year's project, and we are writing for BlackBerry to support five current BlackBerry devices. Because the BlackBerry API is backwards-compatible, future BlackBerry devices should be equally supported with little to no modification. We also plan to engineer the core functionality of the application to be coupled with Java Micro Edition (ME) to make it easy to port to other Java-based mobile device platforms, such as Android.

Requirements

TEAMS-Mobile should support an online mode in which the user logs into the TEAMS-RDS server with their credentials, with the option to remember these credentials for future sessions. Once logged in, the user selects a system to troubleshoot as well as various properties, such as model and serial number. The user cannot bring it seems that the client into offline mode until they have specified symptoms, required resources, and optional resources. It is at this point that the TEAMS-RDS server is ready to return a troubleshooting tree if necessary in the future.

The user then manually specifies when to take troubleshooting into offline mode. Once a session goes into offline mode, it cannot be brought back online by reconnecting to the server. The only further communication with the server at that point is to report back a log file with the troubleshooting results and feedback.

Similar to TEAMATE, individual troubleshooting steps should require that the user reads all applicable notes, cautions, and warnings before continuing with that step. Also, in offline mode, the user should be able to step back and forth through troubleshooting steps. When troubleshooting is complete (whether successful or not), the user must enter feedback for this session. Then a log file of this session is generated and stored locally awaiting transmission back to the TEAMS-RDS server.

The log file will be uploaded to the TEAMS-RDS server across many protocols, such as HTTP(S), FTP, and SMTP. This is configured by the user. Log files should only be synchronized with the server where the troubleshooting started. If a user never brings the troubleshooting session offline then a log should not be generated since TEAMS-RDS will have performed all the logging thus far.

Specifications

TEAMS-Mobile is specified to work on BlackBerry devices with OS version 4.6 and higher. The Java development environment we will use is the BlackBerry JDE 4.6 plugin for the Eclipse 3.4 IDE. Note that the version of the BlackBerry JDE, 4.6, matches the lowest common BlackBerry OS version which is supported.

In addition, network connectivity is required for initial configuration steps between the TEAMS-Mobile client and the TEAMS-RDS server. It is also needed for reporting log files of completed troubleshooting sessions back to the server.

The following is a table of the five specific devices on which we will test TEAMS-Mobile. This selection covers the gamut of modern-day BlackBerry models, from trackball devices with various screen sizes up to the Storm with its touchscreen interface. We do not have the physical devices on hand to test, so we will use simulators provided by RIM.

Model	Display	Keyboard	Interface	Processor	RAM
Curve 8900	480 × 360	Hard	Trackball	512 MHz	256 MB flash
Bold 9000	480 × 320	Hard	Trackball	624 MHz	128 MB flash
Tour 9630	480 × 360	Hard	Trackball	528 MHz	256 MB flash
Storm 9500/9530	360 × 480	Soft	Touchscreen	528 MHz	128 MB flash

Design Choices

Mobile Platform

BlackBerry and iPhone are two of the most popular smartphone brands on the market today, they have certainly outpaced windows mobile devices and most likely will continue to be the most popular consumer choices, besides potentially Android phones (Android also runs on the java framework like the BlackBerry, so developing for BlackBerry will ease a port to Android). Even though BlackBerry and iPhone may be nearly as popular as each other in a consumer market, it seems that Blackberry is often preferred over iPhone in the business setting. Many companies own BlackBerry servers, which ensures that employees will buy BlackBerry devices instead of switching to other alternatives, and maintaining BlackBerry popularity.

The development environment for the BlackBerry device was also a better choice than that of the iPhone. The iPhone would have required a usable Apple machine to develop on, which was not available and would have needed to be purchased. All BlackBerry tools were open source and freeware which has made it easier and far more cost effective to develop the blackberry device. Especially since our budget is zero dollars.

Blackberry JDE Version (4.5.0)

We decided to support the BlackBerry JDE version 4.5.0 and up, on the basis that it is compatible with many devices on the market today. Specifically, we are supporting the Curve 8900, Bold 9000, Tour 9630, and the Storm 9500 and 9530. The JDE version number corresponds to the version number of the BlackBerry operating system running on these devices. It is designed to be fully backwards-compatible, so that any future operating systems should be able to support our application with no modifications.

The newest JDE is version 5.0.0 and is in beta testing right now. It shifts the navigation paradigm away from trackballs and towards touchscreen devices with large displays and soft keyboards. The newer JDE also features nicer debugging tools, including hot-swapping code on a running simulator. Once the market shifts toward newer devices, it may be beneficial to update TEAMS-Mobile to support version 5.0.0.

Online and Offline Mode

TEAMS-RDS already has an online troubleshooting site that is specifically formatted for mobile Web browsers. It is through a Web browser that the user first begins their troubleshooting session. After supplying configuration options to define the address of the server, the user can then log in, select a model, symptoms, resources, and so on, and may then begin troubleshooting directly in the browser.

At this point, the user may choose to go offline. The TEAMS-Mobile client will send an HTTP request to the server to determine if it can take the session offline, and to retrieve a static diagnostic tree if applicable. Then the ModelParser class will be used to parse the XML tree and begin logging the user's session. It is at this point that the interface switches to native BlackBerry GUI elements.

XML Parser

We looked at a number of XML parsers, including Oracle's parser, Apache's Xerxes, Sun's JAXP and JDOM, among others. For ease of porting from last year's incarnation of TEAMS-Mobile, we decided to use a DOM parser as opposed to a SAX parser. Since the minimal DOM parser included in the BlackBerry API did not include the ability to use XPath, an ability we need for our project, we decided to find a third party parser. Many of the standard parsing solutions carried far too large a footprint to be justified for our mobile platform, so we decided on a lightweight DOM parser called Sparta that was made specifically for Java ME applications.

Code Walkthrough

Namespaces

com.hp.hpl.sparta	Contains third-party Sparta XML DOM parser implementation
com.hp.hpl.sparta.xpath	Contains third-part Sparta XPath implementation
teamsmobile	Initializes application and defines access to global variablesthat
teamsmobile.config	Stores and retrieves application-wide user preferences
teamsmobile.resources	Provides string tables for internationalization
teamsmobile.session	Contains objects that represent a troubleshooting session and all its individual types of steps
teamsmobile.session.logging	Provides one-to-one mapping of Java business objects to the log XML format
teamsmobile.session.logging.transport	Defines different mechanisms by which a log can be submitted to the server
teamsmobile.ui	Provides miscellaneous custom UI components
teamsmobile.ui.browser	Provides a simple implementation of an embedded Web browser
teamsmobile.ui.dialogs	Defines all custom modal dialogs
teamsmobile.ui.screens	Defines the GUI implementation of all the main screens the user interacts with
teamsmobile.ui.screens.config	Defines the main configuration screen and any sub-screens
teamsmobile.util	Provides generic functionality for miscellaneous classes
teamsmobile.xml	Provides logic to parse and log interaction with a static XML decision tree

Important Components

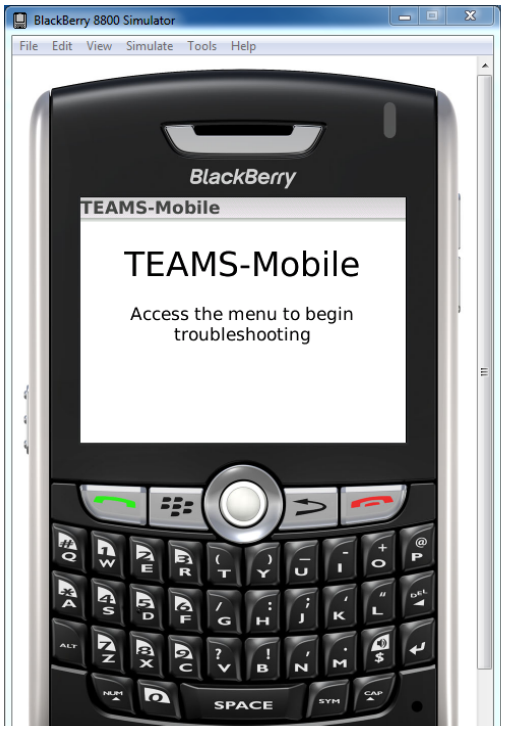
Arguably, the core component of this project is *teamsmobile.xml.ModelParser*. This is a near-direct port of the model parser class from the Windows Mobile project. This class traverses any static XML decision tree and determines the flow of the offline troubleshooting session. It is contained within an instance of the *teamsmobile.session.Session* class, which encapsulates the functionality of the parser into a user-friendly interface. The *Session* class also keeps an instance of *teamsmobile.session.logging.SessionLog* to compile a log of all the user's actions for transmission back to the server.

The log file is submitted by any implementer of the abstract class *teamsmobile.session.logging.transport.TransportMethod*. This class exposes a simple public routine, *send(InputStream, String)*, that handles the entire sending of the log XML file. *EmailTransport* is one such implementation.

All of the application configuration settings are collected in the *teamsmobile.config.Settings* class, which contains public functions for loading and saving settings from a persistent object store provided by the BlackBerry operating system.

Implementation

As stated earlier, the Java development environment we will use is the BlackBerry JDE 4.5.0 plugin for the Eclipse 3.4 IDE. Java is also used by other phone manufacturers for their devices, such as Google (Android), LG, Palm, and Nokia. The development environment includes BlackBerry device simulators, so we can test our system without requiring the purchase of any physical devices. An image of one of the simulators running our project in offline mode is shown below:

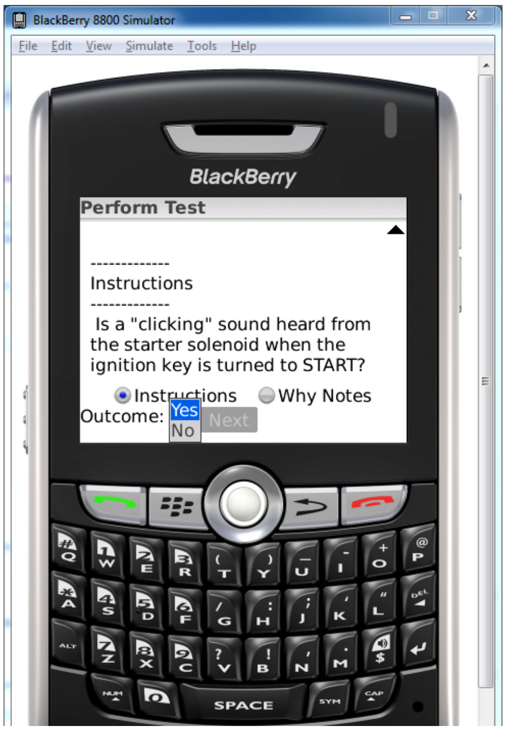


TEAMS-Mobile

TEAMS-Mobile

Access the menu to begin troubleshooting

The welcome screen



Perform Test

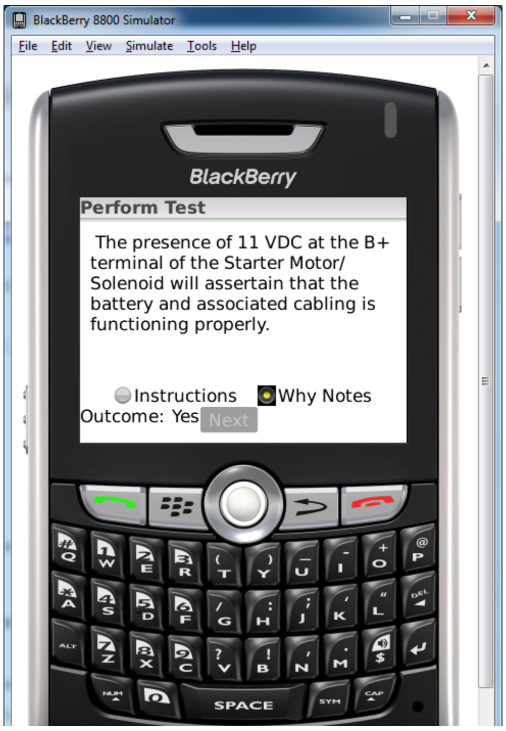
Instructions

Is a "clicking" sound heard from the starter solenoid when the ignition key is turned to START?

● Instructions ○ Why Notes

Outcome: Yes No Next

A test step with two possible outcomes



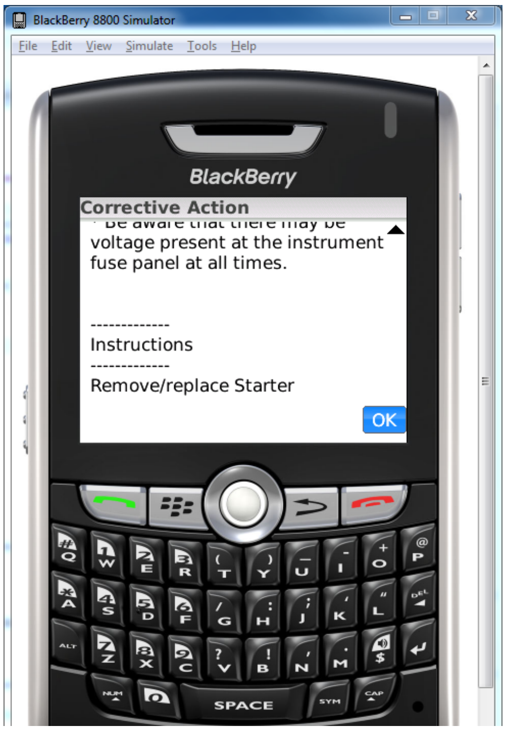
Perform Test

The presence of 11 VDC at the B+ terminal of the Starter Motor/ Solenoid will assertain that the battery and associated cabling is functioning properly.

○ Instructions ● Why Notes

Outcome: Yes Next

A test step showing the "why" notes



Corrective Action

Be aware that there may be voltage present at the instrument fuse panel at all times.

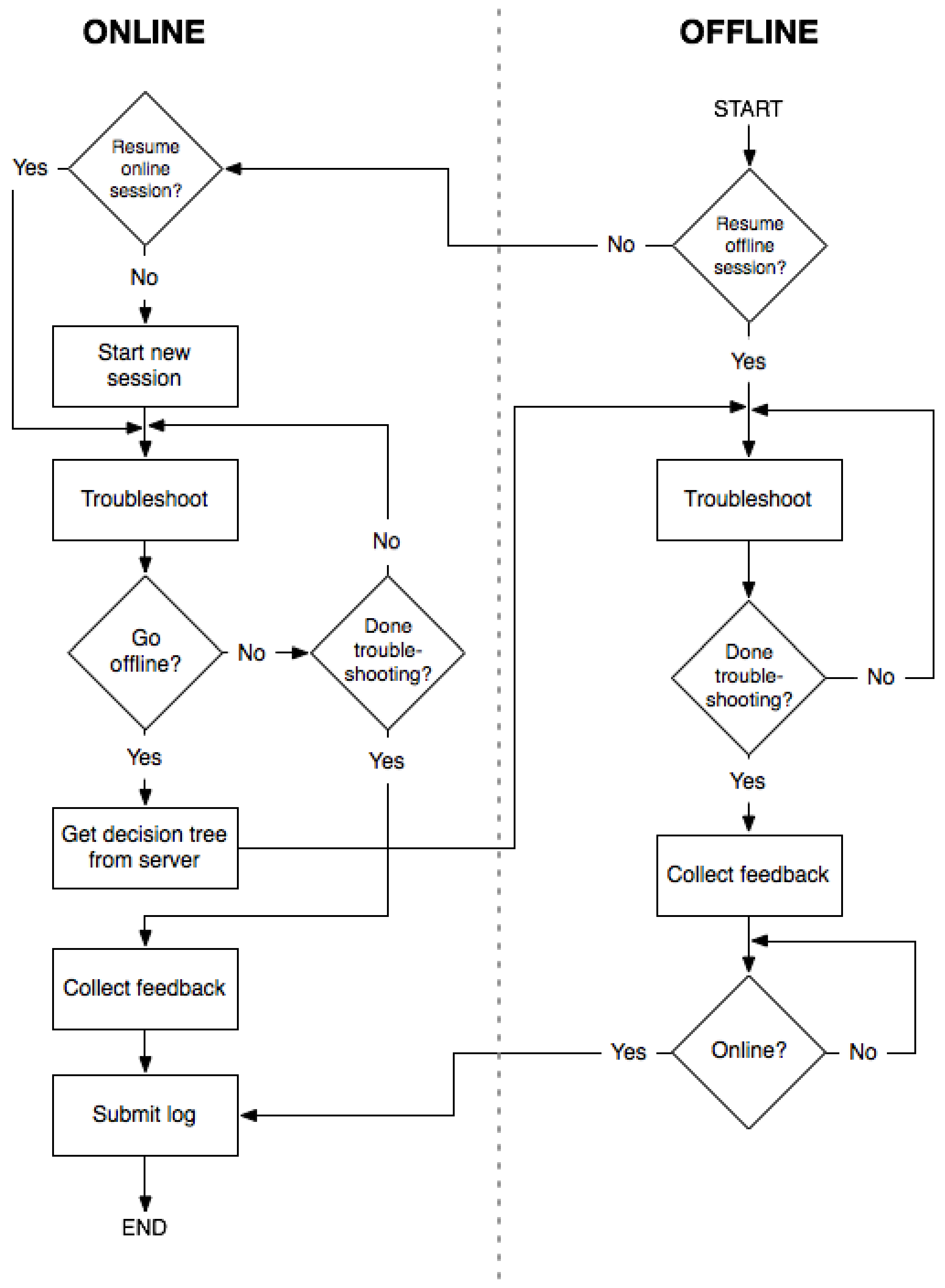
Instructions

Remove/replace Starter

OK

The final step, a corrective action

A key functionality of the system is the transition between online and offline modes, as well as the offline logging functionality with the ability to report back to the RDS server. A diagram of the program flow between online and offline modes is shown below:



Online Component

Unfortunately, the RIM BlackBerry API does not contain an easy-to-use browser field to add to one's applications. Instead, there are a myriad of classes in the net.rim.device.api.browser.field package to facilitate the rendering and event-handling of the browser. This means that basic browser functionality such as history, cookies, and caching must be handled by the implementer. The solution was to use some provided sample code that implemented a single-threaded browser very simply. This code was encapsulated in the teamsmobile.ui.browser package for convenience.

The menu for the browser screen was modified to include a "Go Offline" menu item. This menu item requests a specific URL from TEAMS-RDS and attempts to download a static XML decision tree to continue the troubleshooting session offline. If the server determines that it cannot go offline at that time, it will return some type of error response not yet determined, and then the client should inform the user of this fact and continue in the online mode.

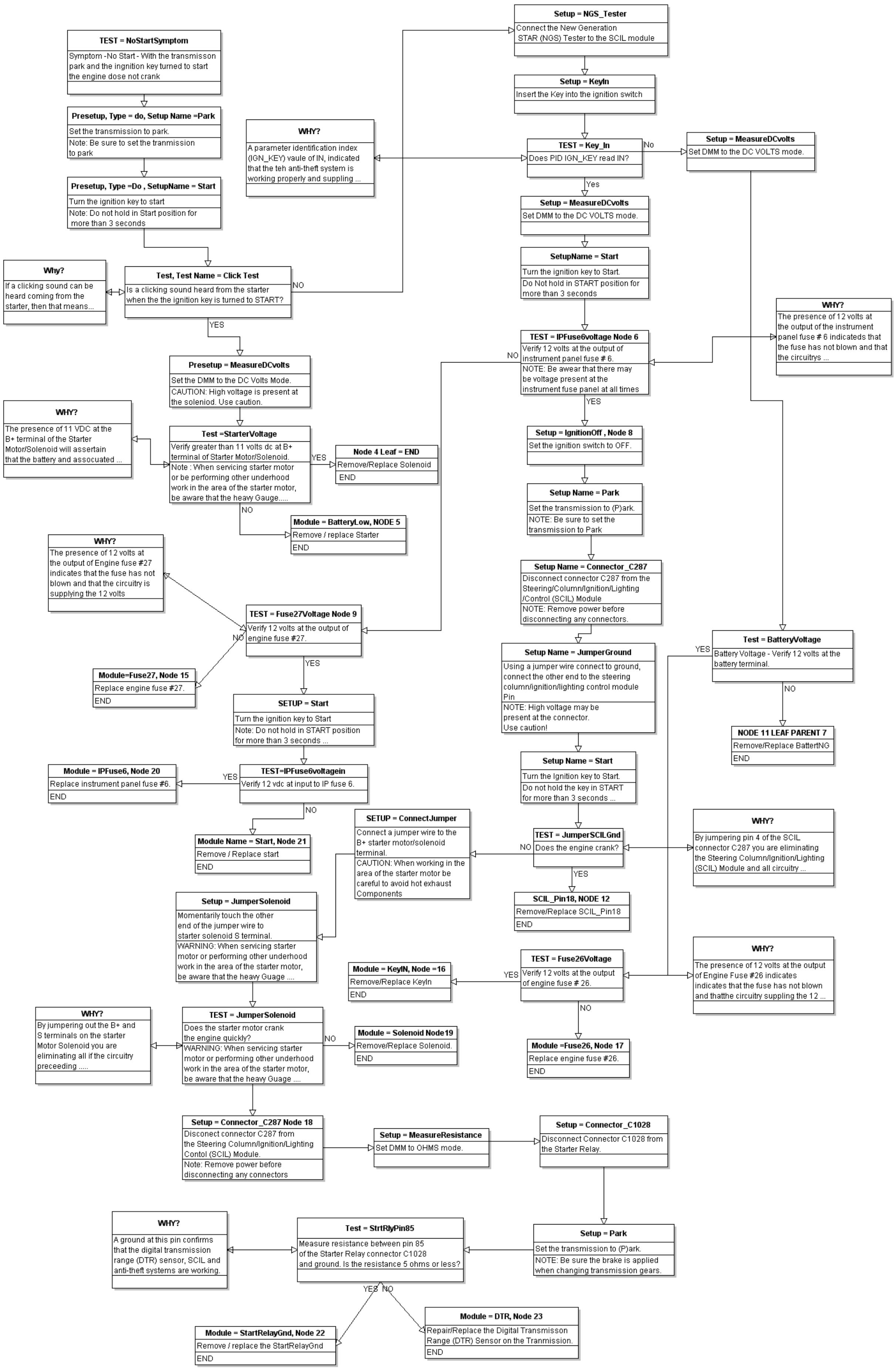
Offline Component

The offline component is entirely a native BlackBerry application. After receiving a diagnostic tree from RDS, the application is able to parse the XML and walk the user through the troubleshooting process while logging pertinent information on the session as per QSI's logging format.

The diagnostic tree received from RDS consists of four main node types: pre-setup, test, post-setup, and module (leaf). When a user starts a session the application finds the first test to be performed. Upon completion of the pre-setup for that test, the application presents the user with the test instructions. A test node may contain sub-nodes of advisories, warning the user of potential injury during the test, and why notes, which explain to the user why the current test is pertinent. After completing the test the application moves to any necessary post-setup, which may be nothing, and according to the user input on the result of the test then moves to the next test or to a module node. Module nodes are leaf nodes which represent the conclusion of a troubleshooting session. Upon reaching a module node, the application presents the user with a corrective action screen and then asks for feedback on the troubleshooting session. A diagram of the diagnostic tree flow is on the next page.

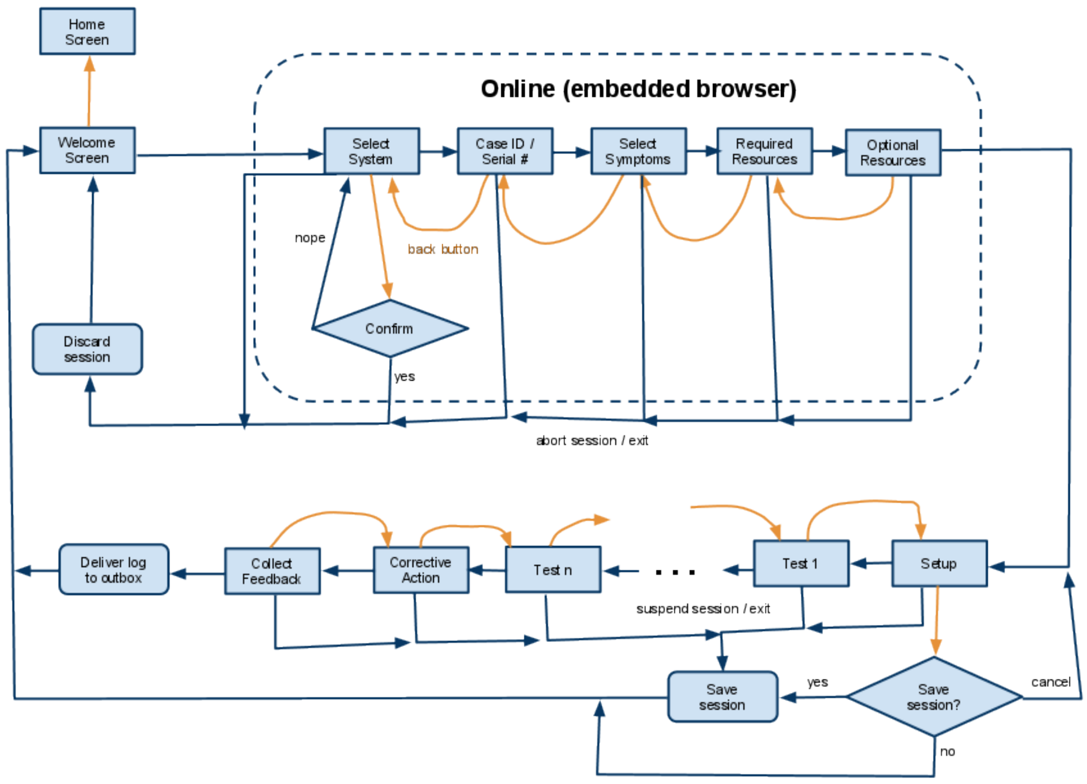
While guiding the user through the diagnostic tree, the application keeps track of information on what the user does and the outcome of the troubleshooting session. Upon completion of the corrective action, the application generates a new XML file that follows the database table format used in QSI's logging system and fits all of the requirements for being sent to and read by RDS. The XML log file is then placed in the BlackBerry's outbox to be automatically sent to QSI when network connectivity is available.
A simplified template of the xml log file format follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<RDSLOGS VERSION="1.1"
  MODEL_NAME="<model-name>_<model-revision>"
  SYSTEM_MODEL_NAME="<model-name>"
  SYSTEM_MODEL_REVISION="<model-revision>"
  SYSTEM_NAME="<system-name>"
  SYSTEM_REVISION="<system-revision>"
  SESSION_ID="<session-id>"
  START_TIME="<session-start-time>"
  USER_NAME="<user>">
  <TABLE NAME="<table-name>" NUM_COLUMNS="<table-numcolumns>">
    <ROW>
      <COLUMN NAME="<column-name>" NULL="[true|false]" TYPE="
       [DOUBLE|BIGINT|INTEGER|SMALLINT|VARCHAR|
       LONGVARCHAR|TIMESTAMP|DATE|LONGVARBINARY]" >
                <![CDATA[<column-value>]]></COLUMN>
      <COLUMN ...>...</COLUMN>
    </ROW>     <ROW>  ...</ROW>    </TABLE>    <TABLE ...> ... </TABLE>     </RDSLOGS>
```

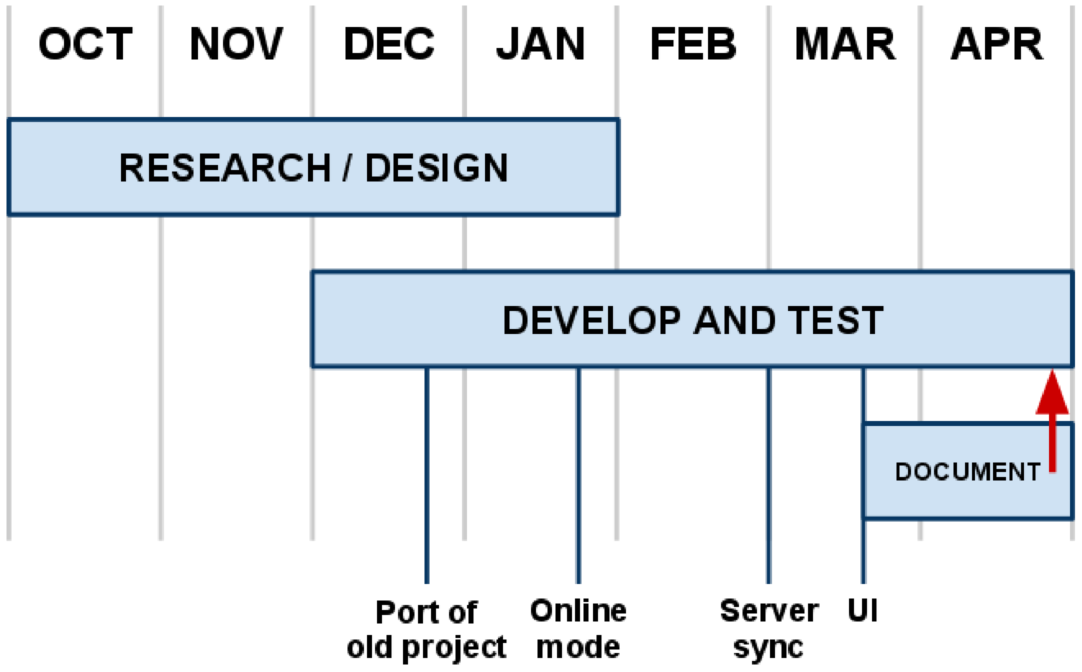

TEST = NoStartSymptom
Symptom -No Start - With the transmisson park and the ingnition key turned to start the engine dose not crank
Presetup, Type = do, Setup Name =Park
Set the transmission to park.
Note: Be sure to set the tranmission to park
Presetup, Type =Do , SetupName = Start
Turn the ignition key to start
Note: Do not hold in Start position for more than 3 seconds
Why?
If a clicking sound can be heard coming from the starter, then that means...
Test, Test Name = Click Test
Is a clicking sound heard from the starter when the the ignition key is turned to START?
NO
YES
WHY?
A parameter identification index (IGN_KEY) vaule of IN, indicated that the teh anti-theft system is working properly and suppling ...
Setup = NGS_Tester
Connect the New Generation STAR (NGS) Tester to the SCIL module
Setup = KeyIn
Insert the Key into the ignition switch
TEST = Key_In
Does PID IGN_KEY read IN?
No
Setup = MeasureDCvolts
Set DMM to the DC VOLTS mode.
Yes
Setup = MeasureDCvolts
Set DMM to the DC VOLTS mode.
SetupName = Start
Turn the ignition key to Start.
Do Not hold in START position for more than 3 seconds
WHY?
The presence of 12 volts at the output of the instrument panel fuse # 6 indicated that the fuse has not blown and that the circuitrys ...
NO
TEST = IPFuse6voltage Node 6
Verify 12 volts at the output of instrument panel fuse # 6.
NOTE: Be awear that there may be voltage present at the instrument fuse panel at all times
YES
Presetup = MeasureDCvolts
Set the DMM to the DC Volts Mode.
CAUTION: High voltage is present at the soleniod. Use caution.
WHY?
The presence of 11 VDC at the B+ terminal of the Starter Motor/Solenoid will assertain that the battery and assocuated ...
Test =StarterVoltage
Verify greater than 11 volts dc at B+ terminal of Starter Motor/Solenoid.
Note : When servicing starter motor or be performing other underhood work in the area of the starter motor, be aware that the heavy Gauge.....
YES
Node 4 Leaf = END
Remove/Replace Solenoid
END
NO
Module = BatteryLow, NODE 5
Remove / replace Starter
END
Setup = IgnitionOff , Node 8
Set the ignition switch to OFF.
Setup Name = Park
Set the transmission to (P)ark.
NOTE: Be sure to set the transmission to Park
Setup Name = Connector_C287
Disconnect connector C287 from the Steering/Column/Ignition/Lighting /Control (SCIL) Module
NOTE: Remove power before disconnecting any connectors.
WHY?
The presence of 12 volts at the output of Engine fuse #27 indicates that the fuse has not blown and that the circuitry is supplying the 12 volts
TEST = Fuse27Voltage Node 9
Verify 12 volts at the output of engine fuse #27.
NO
Module=Fuse27, Node 15
Replace engine fuse #27.
END
YES
SETUP = Start
Turn the ignition key to Start
Note: Do not hold in START position for more than 3 seconds ...
Setup Name = JumperGround
Using a jumper wire connect to ground, connect the other end to the steering column/ignition/lighting control module Pin
NOTE: High voltage may be present at the connector.
Use caution!
YES
Test = BatteryVoltage
Battery Voltage - Verify 12 volts at the battery terminal.
NO
NODE 11 LEAF PARENT 7
Remove/Replace BattertNG
END
Module = IPFuse6, Node 20
Replace instrument panel fuse #6.
END
YES
TEST=IPFuse6voltagein
Verify 12 vdc at input to IP fuse 6.
NO
Setup Name = Start
Turn the Ignition key to Start.
Do not hold the key in START for more than 3 seconds ...
Module Name = Start, Node 21
Remove / Replace start
END
SETUP = ConnectJumper
Connect a jumper wire to the B+ starter motor/solenoid terminal.
CAUTION: When working in the area of the starter motor be careful to avoid hot exhaust Components
NO
TEST = JumperSCILGnd
Does the engine crank?
YES
WHY?
By jumpering pin 4 of the SCIL connector C287 you are eliminating the Steering Column/Ignition/Lighting (SCIL) Module and all circuitry ...
SCIL_Pin18, NODE 12
Remove/Replace SCIL_Pin18
END
Setup = JumperSolenoid
Momentarily touch the other end of the jumper wire to starter solenoid S terminal.
WARNING: When servicing starter motor or performing other underhood work in the area of the starter motor, be aware that the heavy Guage ....
Module = KeyIN, Node =16
Remove/Replace KeyIn
END
YES
TEST = Fuse26Voltage
Verify 12 volts at the output of engine fuse # 26.
WHY?
The presence of 12 volts at the output of Engine Fuse #26 indicates indicates that the fuse has not blown and thatthe circuitry suppling the 12 ...
NO
Module =Fuse26, Node 17
Replace engine fuse #26.
END
WHY?
By jumpering out the B+ and S terminals on the starter Motor Solenoid you are eliminating all if the circuitry preceeding .....
TEST = JumperSolenoid
Does the starter motor crank the engine quickly?
WARNING: When servicing starter motor or performing other underhood work in the area of the starter motor, be aware that the heavy Guage ....
NO
Module = Solenoid Node19
Remove/Replace Solenoid.
END
Setup = Connector_C287 Node 18
Disconnect connector C287 from the Steering Column/Ignition/Lighting Contol (SCIL) Module.
Note: Remove power before disconnecting any connectors
Setup = MeasureResistance
Set DMM to OHMS mode.
Setup = Connector_C1028
Disconnect Connector C1028 from the Starter Relay.
WHY?
A ground at this pin confirms that the digital transmission range (DTR) sensor, SCIL and anti-theft systems are working.
Test = StrtRlyPin85
Measure resistance between pin 85 of the Starter Relay connector C1028 and ground. Is the resistance 5 ohms or less?
Setup = Park
Set the transmission to (P)ark.
NOTE: Be sure the brake is applied when changing transmission gears.
YES
NO
Module = StartRelayGnd, Node 22
Remove / replace the StartRelayGnd
END
Module = DTR, Node 23
Repair/Replace the Digital Transmisson Range (DTR) Sensor on the Tranmission.
END

Screen Flow Diagram

This diagram shows the flow of the program as it appears to the end user. Blue arrows indicate going forward in the troubleshooting process, while orange arrows indicate going back, usually implemented when the user presses the "Escape" hardware button on the BlackBerry. The online-only portion of the user session is outlined in the dashed box. For steps outside this box, the user may choose to go into offline mode. It is important to note that the user is not able to go back from the initial troubleshooting step where they went offline. At this step, they may only suspend the session on the BlackBerry device for resuming later.



Timeline



Our first phase, Research and Design, is to gather experience with QSI products as well as research our target platforms and decide on specific specifications and requirements. This design phase has been extended further to overlap with the development phase, because there are many software-level design decisions that have yet to be made. However, they must be made concreted before development can finish, so that we are not trying to hit a moving target, so to speak.

Actual development began withn porting the existing TEAMS-Mobile application from the .NET Compact Framework over to Java ME and the BlackBerry SDK. In this phase we learned the XML decision tree format better and got a feeling for BlackBerry development.

We then wrote a layer of the application to communicate with the server to act in "online mode" by implementing a bare-bones web browser. Then the offline logging and transportation of log files functionality was completed and tested. Finally, the user interface (UI) was evaluated and tweaked to specification.

Budget

All the BlackBerry development tools are provided as freeware by RIM, and the Eclipse IDE is open-source. Our development server was provided by the School of Engineering, and we're using the freeware Subversion server VisualSVN. Therefore, we did not have any expenses, and our total budget was $0.

Future Considerations

Online to Offline Transition

There are a few implementation details left in the transition of the client troubleshooting session from the online mode to the offline mode. First, the client must be able to properly consume the HTTP response of RDS to determine if it can go offline.

As noted before, there is a lot of logging information that the client needs that is not currently being delivered from RDS. Once Qualtech adds this additional metadata to the XML decision tree, it is job of the BlackBerry application to parse out this additional information and add relevant session information to the output log file.

Graceful Exception Handling

Many of the handled exceptions in our application simply print the exception text to the debugging console and then halt. These exceptions should be handled more gracefully, sending a dialog to the user, and then transitioning the troubleshooting session to the appropriate state and displaying the appropriate screen following the exception.

Polishing of the GUI

The BlackBerry graphical user interface is usable, but could use some tweaks to achieve a more pleasing, intuitive layout. Additionally, the XSL stylesheets that define the Web interface of TEAMATE could be tweaked to better match the user interface components of the BlackBerry platform.

Additional Transport Mechanisms

TEAMS-Mobile currently only supports the submission of log files to RDS via a plain-text, unencrypted e-mail message. Additional steps should be taken to protect this log file from tampering by the user. We would also like to support additional methods to submit the log to RDS, such as HTTP and FTP. See the Requirements appendix for more information.

Saving of Session State

Currently there is no mechanism for TEAMS-Mobile to save the state of a session that has been taken offline. These session objects and their logs should be saved in the application-wide configuration object for an easy implementation. When a session log is sent to RDS, its session should then be removed from this local session store.

Appendix: TEAMS-Mobile Requirements

Online Mode

- Configuration screen should be available before logging in to let user change server address, reporting method, etc.
- User must be able to log in as a technician and select and existing system model and serial number.
    - User should be able to save login credentials
- Users cannot bring the troubleshooting session offline until they have selected symptom(s), required resource(s) and optional resource(s)

Offline Mode

- Once a troubleshooting session is brought offline it cannot be brought back online
- If the application quits in the middle of a troubleshooting session, it should be brought offline and the user should be given the option to continue that session when restarting the application

Troubleshooting

- Individual troubleshooting steps should require that the user read notes, cautions, and warning messages before continuing with that step.
- User should be able to go back or forward through troubleshooting steps (offline only)
- At the end of a troubleshooting session a user must enter feedback for the session
- User should be able to view list of session history
- Progress bar should display user's progression in the troubleshooting session

Log Files

- Once the troubleshooting session is complete a log will be generated in a format that TEAMS-RDS can consume
- Log will be pushed to TEAMS-RDS using the method that has been configured by the user. If no configuration has been specified the log will go into a temporary holding store.
    - The temporary holding store must be secure so the user cannot modify the test results.
- Logs can be sent to TEAMS-RDS using the following methods:
    - HTTP/HTTPS
    - FTP
    - File (more than likely network push)
    - Email
- Logs should only be synchronized with the TEAMS-RDS server upon which the session was started
- Logs should be automatically synchronized with the TEAMS-RDS server as soon as possible, and without user intervention.
- If a user never brings the troubleshooting session offline then a log should not be generated since TEAMS-RDS has performed all the logging thus far.

User Interface Screens

    - Welcome (determine what to do i.e. start session, configure application)
    - Configuration
    - Login (to TEAMS-RDS)
    - Select a system
    - Select a configuration (if the system selected is a configurable system)
    - Pre-session Information Entry (serial number & case id)
    - Select a symptom(s)
    - Select required resource(s)
    - Select optional resource(s)
    - Pre-setup
    - Post-setup
    - Test
    - Multi-outcome test
    - Corrective Action
    - Feedback / Submit Log Form
- UI screens should scale appropriately for multiple BlackBerry devices (different resolutions and touch screen / trackball)

Logging

- Menu Option for going offline when applicable ie. when the session has been started
    - when clicking on offline button, a web request will be made to a specific RDS URL
        - of the form: <RDS URL>/ teamate?action=WORK_OFFLINE&showXML=true
- Log file must be immediately transported to the RDS server; cannot quit until log is sent
- Use e-mail outbox if available
- No user interaction required to synchronize logs; log is sent automatically at end of session.

Configurable application-wide options

- General Options
    - RDS address (validated URL)
    - location to store log files temporarily
- Email
    - address and name to send log files to
    - subject of email
    - body of email

Appendix: TEAMS-Mobile Specifications

Software

- BlackBerry OS v4.5+ (BlackBerry JDE 4.5)
- TEAMS-RDS v11.1.3+

Hardware

- TEAMS-RDS server
  - OS: Windows Server 2003 & 2008, Solaris 9/10, RHEL 4/5
  - Connectivity limited by capacity of BlackBerry.

- BlackBerry mobile device
  - Supported devices with BlackBerry OS version 4.6+
    - Curve 8900
    - Bold 9000
    - Tour 9630
    - Storm 9500
    - Storm 9530
  - Connectivity: cell and/or WiFi b/g

Model	Display	Keyboard	Interface	Processor	RAM
Curve 8900	480 × 360	Hard	Trackball	512 MHz	256 MB flash
Bold 9000	480 × 320	Hard	Trackball	624 MHz	128 MB flash
Tour 9630	480 × 360	Hard	Trackball	528 MHz	256 MB flash
Storm 9500/9530	360 × 480	Soft	Touchscreen	528 MHz	128 MB flash