COMP 4480 – Directed Studies – Final Report

Stefan Schielke / T00226751

12/22/2014

Comp 4480 / Directed Studies / Kevin O'Neil / Fall 2014 / Thompson Rivers University

Table Of Contents

BB10 PhoneGap Project

About UCOSP

Undergraduate Capstone Open Source Projects (UCOSP) brings together students from coast to coast to work together on joint capstone projects. Students learn first-hand what distributed development is like. Each team has students from two to four universities, and uses a mix of agile and open source processes under the supervision of a faculty or industry lead. (UCOSP - About)

Why me

This course is used as a replacement for a lower level business elective that goes towards my Bachelors of Business Administration degree. I had a remaining 'Social Science' elective that I had yet to fulfil. I spoke with Barb Pillar - SOBE advisor, and explained that I felt that a course that carried more interest and / or relevance would be more beneficial to me in my future aspirations after university. With my standing of 'Mature Student', she agreed and we discussed what options I would like. I explained that I would like to do a directed studies course in Computer Science, Comp 4480, as I felt it could further boost me ahead in my learning. She approved the course and then it was up to me to decide on an instructor and project.

When Kevin O'Neil heard that I was interested in a direct studies course, he approached me and talked to me about UCOSP. I was aware of the course as I know a student that had done it in the previous semester. I felt that this was the perfect opportunity for me to try and bridge the gap between what is taught at University and what happens out in the real world of Open-Source projects. Having the opportunity to work with industry mentors while learning first-hand what distributed development is actually like, and at the same time getting University credit for it. What more could I ask for? How about the additional bonus that there was discussion of a partnership with Facebook Open Academy? My application was submitted at the end of May. There would only be forty students from Canada that would be selected for the fall semester.

I crossed my fingers and waited…

Mid July I heard back from Michelle Craig, chair of the UCOSP Steering Committee that I had been accepted! She also informed us that the partnership with Facebook Open Academy had also not worked out and that we should be holding our code sprint in Toronto at a location still yet to be determined.

There were a few emails at the end of July that were formality, signing our commitment, requesting projects, and arranging flight to the code sprint. By mid-August, we were informed of our assigned project. Within a couple of days we had our first contact from our mentor. Then by the beginning of the semester we had our first instructions for steps to take prior to the code sprint.

And so it all began…

Participating Universities

Brock University
Dalhousie University
Laurentian University
Simon Fraser University
The University of Western Ontario
Thompson Rivers University
University of Alberta
University of British Columbia
University of Guelph
University of Manitoba
University of New Brunswick
University of Ottawa
University of Regina
University of Toronto Mississauga
University of Waterloo
York University

Projects Available

Markus:	Web-based grading platform
UMPLE:	UML modeling and programming tool
BB10 PhoneGap:	Mobile app platform
Formulize:	Database, reporting and workflow management system
Eclipse Orion:	Integrated development environment
Review Board:	Code review tool
Freeseer:	Screen-casting tool
Waterbear:	Online development environment

Project Selection

Each student was asked to select their top four projects in order of preference.

My selections:

1. BB10 PhoneGap
2. Formulize
3. Waterbear
4. Review Board

Project Assigned

An email dated 15-August-2014 from Aman Grewal, Department of Computer Science / University of Toronto, confirming our project assignments.

I was pleased that my first selection was accepted and I would be working on the BB10 PhoneGap project. It was not that I am a BlackBerry guy. More that of the projects that were available, this one stood out most mainly due to it being mobile related. Since I had already done two courses in Android development, I felt it was a natural progression for my learning, and interest.

Project Details

BlackBerry is embracing application development with HTML5 and seeking to push the boundaries of what can be accomplished with web technologies on a mobile device. Our goal is to be the premier platform for the mobile web, with the most compliant, high performance, hardware accelerated engine we can create. We are rounding out the development experience with emulators, simulators, live Web Inspector debugging, support for all major frameworks, and we're doing all of this in the open on Github.com under the Apache 2 license.

This term BlackBerry is focused on PhoneGap/Cordova compatibility and our next version of the WebWorks SDK will be powered by Cordova. With the open source community we will be working on porting native plugins from iOS and Android, and writing new ones for BlackBerry 10. Students should be familiar with C++ or JavaScript development, though they do not need to be experts with both. We will work in pairs and teams to round out the necessary skill set. (UCOSP - Projects)

Project Team

Tim Windsor – **Mentor**
Kris Flores – *University of Toronto / past team member* – ***assistant***
Tanya Homynyk – *University of Alberta*
Jim Wen – *University of Alberta*
Justin Carvalho – *University of Guelph*
Stefan Schielke – *Thompson Rivers University*
Tim Tung – *Thompson Rivers University*
Yifan Ren – *University of Waterloo*

About Tim Windsor

Tim received his Bachelor's Degree, Honours BMath in Computer Science from University of Waterloo in 2003. He started working for RIM in January 2007 as a Technical Partnership Manager. In August 2008 he changed to Application Development Consultant for BlackBerry where he stayed until November 2014. He is now officially the Open Source Technical Lead at BlackBerry.
His summary on LinkedIn says:
"BlackBerry Developer Relations since Jan 2007. Experienced in working with top mobile application developers delivering mission critical software for Fortune 500 companies - to helping solo developers

with their first applications. Now working through open source to support my colleagues and other developers with BlackBerry WebWorks samples and extensions, and building the BlackBerry open source community." (Tim Windsor)

Technology to be used

C++ - The native development for BlackBerry is using a program called Momentics (version 2.1.1).
JavaScript – Modifications to the code done in either Momentics or Notepad++
HTML5 – All changes to html files done in Notepad++
CSS3 – Changes if required done in Notepad++
BlackBerry Z10 – Supplied by BlackBerry and used for testing purposes.
BlackBerry Software Release – 10.2
Git 2.2.1 – Used for forking into the BlackBerry repository
GitHub – Desktop GUI for accessing Git

Installation Instructions

WebWorks SDK

https://developer.blackberry.com/html5/downloads/

This is the tools for build WebWorks (HTML5) applications. This is basically just a version of the Apache Cordova tools, similar to PhoneGap and is compatible with it. It supports a way to write native platform code that is accessed through a JavaScript API. This native code is packaged up and called a plugin, and that's what we're going to work on this term.

Momentics NDK

https://developer.blackberry.com/native/downloads/

This is the main set of tools for writing all native code on the BlackBerry 10 platform. It can be used to make full applications, but we're mostly going to write lower level native code.

Code signing keys

https://developer.blackberry.com/native/documentation/cascades/dev/tools/signing_and_publishing.html

The Native tools have a decent wizard interface to register and configure your signing keys. If you didn't get them as part of the installation process, go through that now using the link and get them installed.

Simulator – optional

The simulator can be installed from any of the download pages. You need a pretty decent machine to run it, since it's a VM, and requires VMWare. If you've got a Mac or Linux laptop I don't think the tools are free, and we won't do much development on it, so it's really quite optional.

Main Repo

https://github.com/blackberry/WebWorks-Community-APIs

This repo houses all our plugins for the various versions of the WebWorks platform. Our time is going to be spent working on the ones in the BB10-Cordova folder. You will want to fork this repo on GitHub so you can get the code and make contributions to it.

Software Descriptions

Momentics

The QNX® Momentics® Tool Suite is a comprehensive, Eclipse-based integrated development environment with innovative profiling tools for maximum insight into system behavior. These unique tools give developers at-a-glance views of realtime interactions, memory profiles, and more, enabling shorter debug times and faster time to market. Multi-core specific tools help developers migrate code cleanly from single-core to multi-core systems, and safely optimize performance. (QNX Momentics Tool Suite)

Notepad ++

Notepad++ is a free (as in "free speech" and also as in "free beer") source code editor and Notepad replacement that supports several languages. Running in the MS Windows environment, its use is governed by GPL License.
Based on the powerful editing component Scintilla, Notepad++ is written in C++ and uses pure Win32 API and STL which ensures a higher execution speed and smaller program size. By optimizing as many routines as possible without losing user friendliness, Notepad++ is trying to reduce the world carbon dioxide emissions. When using less CPU power, the PC can throttle down and reduce power consumption, resulting in a greener environment. (Notepad++)

Git

Git is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and has since become the most widely adopted version control system for software development. (Git (software))

GitHub

GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as wikis, task management, and bug tracking and feature requests for every project. (GitHub)

Process of creating a new project

1. From a command prompt, change the directory to the workspace. *Note that there cannot be any spaces in the working directory path.
   "C:\Users\Stefan_Schielke\momentics-workspace" is acceptable.
   "C:\Users\Stefan Schielke\momentics-workspace" is not, and will error out.

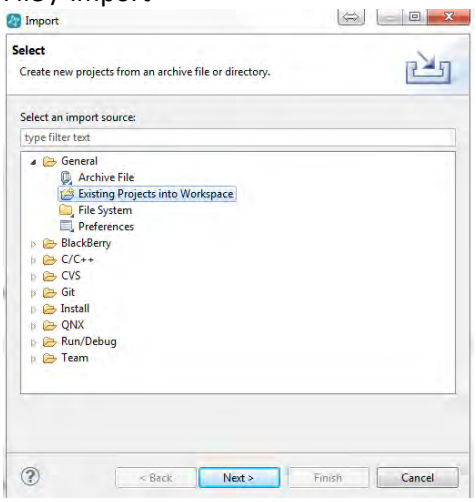
2. Create a new WebWorks project:
   "webworks create <newProject>"

3. Go into the directory of the new project:
   "cd <newProject>"

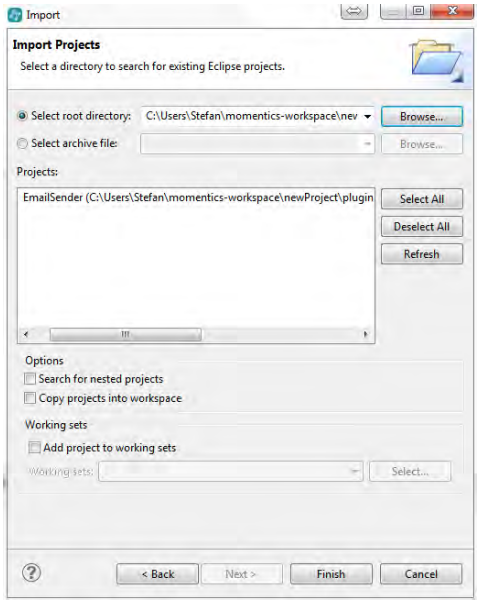
4. For existing projects, there is usually the current plugin and sample plugin in the Git repository.
   a. Copy the "plugin" folder into the current directory.
   b. Copy the www folder from the sample folder into the current directory.

5. Add the plugin to the WebWorks project:
   "webworks plugin add C:\Users\Stefan\momentics-workspace\<newProject>\plugin"

6. Add the project to Momentics:
   File / Import



General / Existing Projects to Workspace
Next

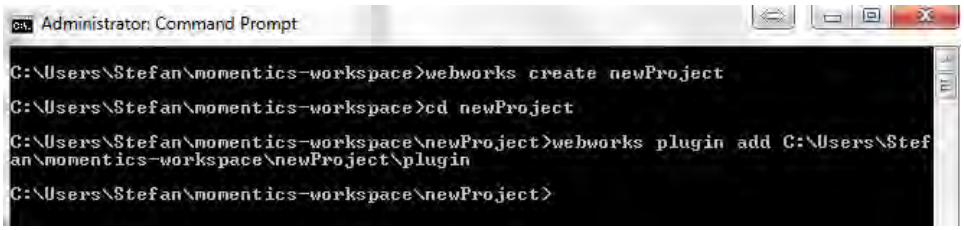


Select Root Directory and browse to the plugin directory (note: not plugins)
Finish

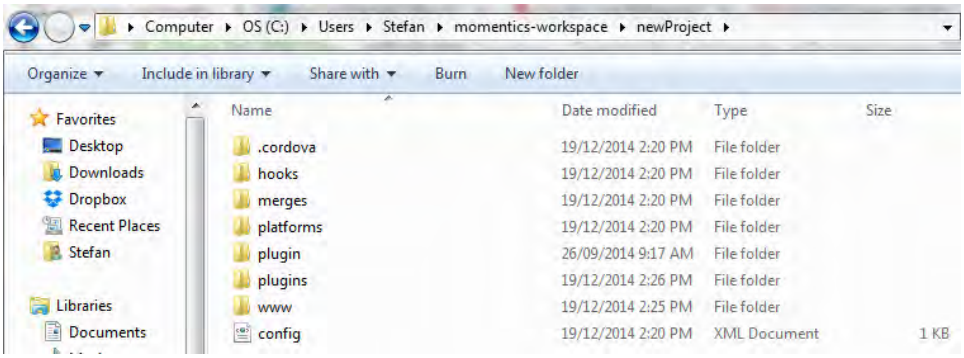
7. To run the project:
   "webworks run –devicepass <device password>"
   (–devicepass allows you to bypass having to enter the password every time you run the project)

8. If any changes are made to any of the .cpp or .js files, the project must be built in Momentics, the plugin removed from the project:
   "webworks plugin remove com.blackberry.community.<pluginName>"
   Steps 5 & 7 to re-add and run again (from the Command Prompt, use the up-arrow to retrieve previously run commands)

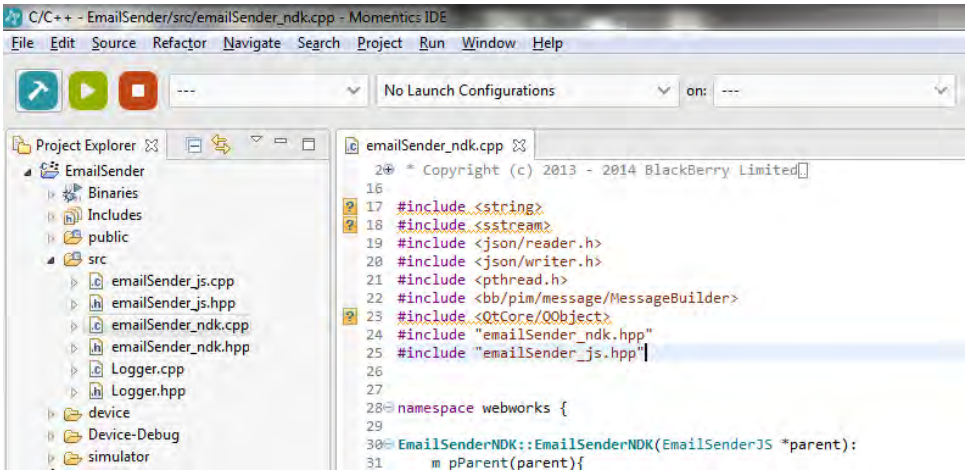
Command Prompt



Window Explorer directory structure



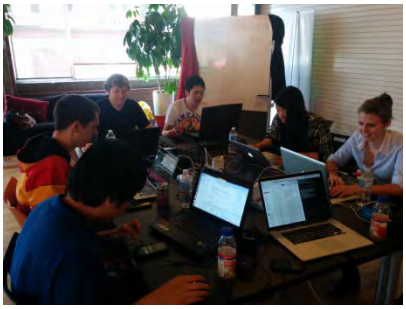
Momentics file structure



Code Sprint

After arriving in Toronto Thursday, after a long day of travel (for some), the students of UCOSP got together at the University of Toronto for a meet and greet. We played a few ice breaker games to help encourage some interaction, led by Karen Reid; had a presentation by Raquel Urtasun who introduced and promoted U of T's graduate program; followed by general mingling.

Friday morning we went to the office of our host, Mozilla. We were given a welcome to the UCOSP by Michelle Craig, and then an overview of Mozilla and their vision by Mike Hoye. We jumped into our teams and introduced ourselves a little more. The PhoneGap team consists of: BlackBerry mentor, Tim Windsor; past UCOSP member and University of Toronto student, Kris; University of Guelph student, Justin; University of Waterloo student, Yifan; University of Alberta students, Tanya and Jim; and Thompson Rivers University students, Stefan and Tim. Here is a photo of the team at work.



During our visit, we were fortunate to be provided a tour of the Mozilla facility and got to see the team members in Toronto at work in their real-world environment. It was quite exciting to see their development process, the development rooms filled with white boards and Post-it notes that outlined their goals and tasks currently in progress or discussion. As we are all potential developers, it was invaluable to see the real-world interactions of the members. Different areas of the office building provided people with the ability to get away from their desks and work in a shared space in the office or to just either relax in a non-work environment or even take up a game of Ping-Pong.

The tasks were divided up by interest, and then the team members started right into them. As we all learned, there is a learning curve with the introduction of a new API as well as the whole development of a new plugin for the BlackBerry 10 platform. Each team member was provided with a new Z10 or Q10 for testing purposes. This will enable us to test the plugins on actual devices and realize the effect of our efforts immediately on the targeted environment.

The initial tasks that we decided upon ranged from finding and fixing a bug in the Bar-code scanner; finish merges for Facebook connect, Google Analytics, and Distimo SDK; support reading audio metadata; and additional support for the EmailSender plugin. At the end of our day on Sunday, we were all getting much more comfortable with the environment, although there are still some areas of the development process that are not 100% clear, this will all be overcome with time and practice. I look forward in getting to understand the processes of our project and am excited as to what the term will bring.

Project Tasks for Fall 2014

Add BlackBerry 10 support to App Preferences API
Add support for BlackBerry 10 to ExternalScreen plugin
Barcode Scanner BB10 - whitespace character issue
Create a Build and Testing framework for plugins and apps
Dialog Extension for WebWorks
EmailSender plugin should support non-file attachments
EmailSender plugin should support HTML emails
ExtractZIPFile crashes when multiple files are provided (and one of them is an incorrect path)
Finish Distimo SDK merge into PhoneGap plugin
Finish Facebook Connect merge into PhoneGap plugin
Finish Google Analytics merge into PhoneGap plugin
LowLatencyAudio issue when using bluetooth earphone
Optimizations to Barcode Scanner
Port UPnP Extension to Cordova Plugin
Porting PhoneGap AudioRecord API to BlackBerry 10
Support Reading Audio Metadata (ID3) from files
Update ExtractZipFile Compress to handle an array of files for compression
Update Curl Plugin to be Asynchronous
UNZIP Community API not unzipping subdirectories
Zip/Unzip Extension

Tasks Assigned / Selected

Tasks were divided up at the code sprint in Toronto.
I thought that I would try and take an easier looking task to start as I thought it would enable me to get a feel for the IDE, process, and new phone (Z10).
My Selection(s):

- EmailSender plugin should support HTML emails
- EmailSender plugin should support non-file attachments

EmailSender – Support HTML emails

Issue

MessageBuilder can take an HTML type, so if we can add a flag to the API for type, we should be able to accept both Plain text and HTML emails in this API.

Details

The original plugin was made with very basic functionality. Upon initialization it should pull all of the accounts on the phone. The user could then select which account to send the email from. Who to send the message to (To, Cc, & Bcc). The subject and the body. The original plugin only had a single option for sending the message body as Plain Text. This could be found on line 106 in the emailSender_ndk.cpp: `builder->body(MessageBody::PlainText,bodyData);`

Original Sample App

Task

To add the functionality to provide the user to choose what format to send the message. The changes for this were quite simple once I was able to get into the phone and grab the accounts.

Steps taken

- Created new accounts in Gmail, Yahoo, and Outlook (previously Hotmail)
- Added all of the accounts to the phone
- Tested sending and receiving of messages on phone
- Read the Readme.md file in the Git repository for the plugin
- Check the files in the Git repository for the plugin
- Created the plugin as per the example above
- Built the plugin and received an error
    - `fatal error: QtCore/qglobal.h: No such file or directory`
    - After reviewing the problem on StackOverflow, there some includes required
        - Right click over your project in Project Explorer and choose Properties
        - Expand the tree to C/C++ General / Paths and Symbols
        - Change the Configuration in the Paths and Symbols frame to [All configurations]
        - Click the Includes tag and select GNU C in the Languages list

- Click Add... and type ${QNX_TARGET}/usr/include/qt4 and press OK
- Click Add... and type ${QNX_TARGET}/usr/include/qt4/QtCore and press OK

- Build was successful after this
- Tried to get the plugin to recognize the accounts
  - This failed for me
  - Spoke with both Tim and Kris while at the code sprint and was told to check the documentation.
  - Looked at the documentation again
  - Still could not figure out why I could not access the accounts
- Added the following changes to the plugin while working on the account issue

index.html

Added a new label for a “Send as:” format type:

```
<label>Send as:</label><br>
    <select id="emailType">
        <option value="html">HTML</option>
        <option value="txt">Plain Text</option>
    </select>
```

index.js

```
"Type": document.getElementById('emailType').value
```

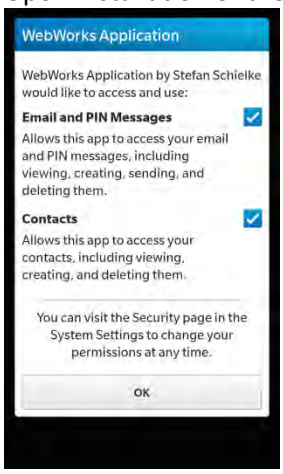
emailSender_ndk.cpp

```
Json::Value Type = input["Type"];
if(Type=="html"){
   builder->body(MessageBody::Html, bodyData);
}
else{
   builder->body(MessageBody::PlainText, bodyData);
}
```

Modified Sample App

- Even though the sample app shows the correct changes, I still could not get it to show the accounts from the phone.
- There are 2 config.xml files that are included in the project structure
  - 1 in the www directory

        - This one contains the permissions that are required to access the contacts & messages on the phone.
    - 1 in the root of the project
        - This one does not automatically contain the permissions
- After adding the permissions to the config.xml in the root, the accounts on the phone would still not appear in the plugin
- I got Tim on a remote access to my system to see if he could show me where the problem still was happening
- After ½ hour of looking it turns out that the build configuration after restarting Momentics changed back to the default of 'Device-Debug'
- Changed this back to just 'Device' and rebuilt
- After removing, re-adding the plugin, and then running the project, success
- Upon installation of the app, it asks for permissions to access *Email and PIN messages* and *Contacts*



- All 3 accounts that were added to the phone, show up in the send from field of the sample app

- Test case 1.1
    - Results

      Gmail (blackberrystefan@gmail.com) – Passed

      Yahoo (stefanblackberry@yahoo.ca) – Passed

      Outlook (blackberrystefan@outlook.com) – Fail
- Cannot send from Outlook account
    - This was a known problem previously
    - Original author also stated that there were issues with Yahoo accounts.

        - This was not a problem now as I am able to send from both Yahoo and Gmail
    - Deleted the Outlook account from the phone
- Test case 1.2
    - Results
      Gmail – Passed
      Yahoo – Passed
- Test case 1.3
    - Results
      Gmail – Passed
      Yahoo – Passed
- Test case 1.4
    - Body of message
      Message with *format*
    - Results
      Gmail – Passed
      Yahoo – Passed
- Test case 1.5
    - Body of message
      Message with format
    - Results
      Gmail – Passed
      Yahoo – Passed
- After successful testing, synch to local GitHub account
- Send Pull Request to Tim for review

Test Cases

1.1 Send from all accounts

Send As: HTML
To: schielke.stefan@gmail.com
From: <see below>
Subject: Test sending message
Body: Test message

1.2 Send as Plain Text format with no formatting

Send As: Plain Text
To: schielke.stefan@gmail.com
From: <see below>
Subject: Test without formatting (Text)
Body: Message with no format

1.3 Send as Html format with no formatting

Send As: HTML
To: schielke.stefan@gmail.com
From: <see below>
Subject: Test without formatting (Html)
Body: Message with no format

1.4 Send as Html format with formatting

Send As: HTML

To: schielke.stefan@gmail.com
From: <see below>
Subject: Test with formatting (Html)
Body: Message with format

1.5 Send as Plain Text format with formatting

Send As: Plain Text
To: schielke.stefan@gmail.com
From: <see below>
Subject: Test with formatting (Text)
Body: Message with format

Associated documentation – BlackBerry Developer – Cascades

Used for obtaining account information off of the phone for use in the app:

Account
https://developer.blackberry.com/native/reference/cascades/app_integration_account.html

AccountService
https://developer.blackberry.com/native/reference/cascades/bb__pim__account__accountservice.html

Service
https://developer.blackberry.com/native/reference/cascades/bb__pim__account__service.html

I would recommend that once version 10.3 is implemented that the account information from the phone be modified to use a different method to pull account information from the phone. Currently the accounts are pulled using a service type of 'Messages'. This will pull just the active accounts from the phone that use a message service.

```
QList<Account> accounts = accountService.accounts(Service::Messages);
```

Once the phones are updated to 10.3 there is another method that can be used to pull all the account information from the phone. This would allow for the emailSender to send from other types of accounts.

QList< Account > **allAccounts** ()
Retrieves the list of Account objects currently stored on the device including any accounts which are disabled. **Return**: Returns a QList containing all Account objects. **Since**: BlackBerry 10.3.0

```
QList<Account> accounts = accountService.allAccounts();
```

Used for sending messages from the app:

Message
https://developer.blackberry.com/native/reference/cascades/app_integration_message.html

MessageBuilder
https://developer.blackberry.com/native/reference/cascades/bb__pim__message__messagebody.html

MessageBody
https://developer.blackberry.com/native/reference/cascades/bb__pim__message__messagebody.html

EmailSender – Support non-file attachments

Issue

MessageBuilder can also take Attachment(s) that can be constructed from strings (or byte arrays):
https://developer.blackberry.com/native/reference/cascades/bb__pim__message__attachment.html#function-attachment-mimetype-name-textdata
Supporting attachments in this way would be beneficial because it would allow webworks apps to append attachments to emails without requiring 'Shared File' permissions (currently the invoke card for email requires a path to the shared directory for attachments; private app storage cannot be used).

Details

Currently the plugin does not have the attachment functionality built into it. The request is to allow for attachments to be added to the plugin and to not to have to set permissions for file access.

Task

To add attachment functionality to the plugin

Steps taken

- First step was to code adding an attachment to the email. I did not make any changes to the .js or .html files at this time as I was just trying to ensure that a message could in fact be added to the email
- There are multiple constructors that can be used to creating an attachment that can be added to a message

Attachment ()
Attachment (const QString &mimeType, const QString &name, const QUrl &path)
Attachment (const QString &mimeType, const QString &name, const QUrl &path, const QVariantMap &metaData)
Attachment (const QString &mimeType, const QString &name, const QString &textData)
Attachment (const QString &mimeType, const QString &name, const QByteArray &data)
Attachment (const QString &mimeType, const QString &name, const QByteArray &data, const QVariantMap &metaData)
Attachment (AttachmentPrivate *attachmentPrivate)
Attachment (const Attachment &other)

- In order to ensure that the plugin could send all different types of messages (.docx, .jpg, .dat, no file extension) I did some extensive research into the QtCore libraries
- For sending an attachment as a byte array, I took the data from the body of the message and converted it into a QByteArray which would then be added as an attachment

```
QByteArray bodyData;
bodyData.append(QString::fromStdString(body.asString()));
```

  Then adds it as an attachment by:

```
Attachment sendAttachment("", "newByteAtt", bodyData);
builder->addAttachment(sendAttachment);
```

  This sends the attachment with a .dat extension (newByteAtt.dat)
- For sending a string (text file) I followed the same process but instead of creating a QByteArray I did it as a QString

```
QString bodyData = QString::fromStdString(body.asString());
Attachment sendAttachment("txt", "newTextAtt", bodyData);
```

```
builder->addAttachment(sendAttachment);
```

This sends the attachment with a .txt extension (newTextAtt.txt)

- Test case 2.1
  - Results
    Gmail – Passed
    Yahoo – Passed
- Test case 2.2
  - Results
    Gmail – Passed
    Yahoo – Passed
- Test case 2.3
  - Results
    Gmail – Passed
    Yahoo – Passed
- Test case 2.4
  - Results
    Gmail – Passed
    Yahoo – Passed
- Next step was to add the ability for the user to specify the path to the file to be added
- I followed in the same format as the initial plugin
- I added a checkbox and a file path location text box to the html

index.html

```
<label><input id="attachment" type="checkbox" value="false">Add attachment</label><br>
<input id="attachmentLocation" type="text" placeholder="/accounts/1000/.../file.ext" />
```

- I added the values to be pulled into the index.js jsonEmail

index.js

```
var jsonEmail =
{
...
   "attachment": document.getElementById('attachment').value,
   "attachmentLocation": document.getElementById('attachmentLocation').value
}
```

- Then I added the functionality to the .cpp file to extract and pull the data
  - I was not satisfied with a hacked solution to extract the file name and extension from a file path. i.e. parsing the string and finding the last instance of '/' and then pulling the full file name using that as a substring to the end of the string. Then finding the last instance of '.' and then getting the extension off of that. I wanted a solution that would work all of the time, not some of the time.
  - If a file was example.tar.gz it would incorrectly have a file name of example.tar and an extension of .gz. This would be incorrect, and writing extensive code to solve this would be both lengthy and time consuming (longer than the semester permits).
  - I looked at available solutions online for this. Most spoke of hacking a solution together, or using an external library such as Boost.
  - I looked deeper into the QtCore documentation and found that there is a class to handle this, QFileInfo. The steps to get the info:
    - Create a QString containing the path
    - Create a QUrl from the QString

- Create a QFileInfo from the QUrl
- Get the file name using method .fileName()
- Get the file extension using method .completeSuffix()

- New include required in the .cpp

```
#include <QFileInfo>
```

- The .cpp file was then ready. I would use the file type, file name, and QUrl to create the attachment (*Attachment (const QString &mimeType, const QString &name, const QUrl &path)*)
- I was passing the file path as "/accounts/1000/shared/documents/example.txt" and adding the file:// inside my .cpp file

emailSender_ndk.cpp

```
if (!msgAttachment.empty()){
  if(msgAttachment.isArray()){
    foreach(Json::Value v, msgAttachment){
      QString qpath = QString::fromStdString("file://" + v.asString());
      QUrl qfilepathq(qpath);
      QFileInfo qfilepath(qpath);
      QString qfilename = qfilepath.fileName();
      QString qfiletype = qfilepath.completeSuffix();
      Attachment msgAttach(qfiletype, qfilename, qfilepathq);
      builder->addAttachment(msgAttach);
    }
  }
  else{
    QString qpath = QString::fromStdString("file://" + msgAttachment.asString());
    QUrl qfilepathq(qpath);
    QFileInfo qfilepath(qpath);
    QString qfilename = qfilepath.fileName();
    QString qfiletype = qfilepath.completeSuffix();
    Attachment msgAttach(qfiletype, qfilename, qfilepathq);
    builder->addAttachment(msgAttach);
  }
}
```

- I decided to not check every case from both accounts; I would be using Gmail only. Also, I would only be testing as Html format type, not from Plain Text.
- Test case 2.5
  - Results

    Gmail – Fail
- Permission missing from the config.xml

config.xml

```
<rim:permit>access_shared</rim:permit>
```

- Test case 2.5
  - Results

    Gmail – Fail
- Ran some debug statements and found that the attachment value has not been set to change to true when the value has been checked
- Added the logic to the .js file to check and validate this

index.js

```
if(document.getElementById('attachment').checked)
  document.getElementById('attachment').value = "true";
else
  document.getElementById('attachment').value = "false";
```

- Test case 2.5
  - Results
    Gmail – Passed
- Test case 2.6
  - Results
    Gmail – Fail
- Running debug statements shows that the multiple files are not processed properly. They are processed as one file path, resulting in the email been queued, but not sent.
- Changed the index.js file to split the pieces so that they can be processed properly. This is not the 'ideal' solution, but it is acceptable for a first release. "It really starts to get tricky with file paths. It's a common area of challenge." (Windsor, RE: Update to emailSender, 2014)

index.js

```
var jsonEmail =
{
…
  "attachmentLocation": document.getElementById('attachmentLocation').value. split(',')
}
```

- Test case 2.6
  - Results
    Gmail – Fail
- The first attachment is processed okay, but the second attachment has a leading space as a result of the split
- Added some logic in the .cpp file to catch this

emailSender_ndk.cpp

```
foreach(Json::Value v, msgAttachment){
  string checkpath = v.asString();
  char ws = ' ';
  checkpath = checkpath.erase(0, checkpath.find_first_not_of(ws));
  checkpath = checkpath.erase(checkpath.find_last_not_of(ws) + 1);
  …
}
```

- Test case 2.6
  - Results
    Gmail – Passed
- Since the data been passed is always an array due to the .split(',), there is no need for an if/else statement as it will always be true if an attachment is selected

emailSender_ndk.cpp

```
if (!attachment.empty()){
  foreach(Json::Value v, attachment){
    // remove any leading or trailing spaces from a file path
    string checkpath = v.asString();
    char ws = ' ';
```

```
            checkpath = checkpath.erase(0, checkpath.find_first_not_of(ws));
            checkpath = checkpath.erase(checkpath.find_last_not_of(ws) + 1);
            // allow for full or partial path to be entered
            string match = "file://";
            if(checkpath.length() >= match.length() && !equal(match.begin(), match.end(),
                                        checkpath.begin()))
               checkpath = "file://" + checkpath;
            QString path = QString::fromStdString(checkpath);
            QUrl filepath(path);
            QFileInfo fileinfo(path);
            QString filename = fileinfo.fileName();
            QString filetype = fileinfo.completeSuffix();
            Attachment msgAttach(filetype, filename, filepath);
            builder->addAttachment(msgAttach);
         }
        }
```

- Code cleaned up (removed debug statements)
- Bought SD Card for testing
- Test case 2.7
    - Results
      Gmail – Passed
- Test case 2.8
    - Results
      Gmail – Passed
- Additional testing using Yahoo! account and Plain Text format
- Test cases 2.5 – 2.8
    - Results
      Yahoo – Passed
- Synch to local GitHub account
- Send Pull Request to Tim for review

Test Cases

2.1 Send as a byte array attachment (Html)

Send As: HTML
To: schielke.stefan@gmail.com
From: <see testing>
Subject: Test Byte Array
Body: This is the data to be sent as a byte array

2.2 Send as a byte array attachment (Plain Text)

Send As: Plain Text
To: schielke.stefan@gmail.com
From: <see testing >
Subject: Test Byte Array
Body: This is the data to be sent as a byte array

2.3 Send as a text attachment (Html)

Send As: HTML
To: schielke.stefan@gmail.com

From: <see testing >
Subject: Test
Body: This is the data to be sent as a text file

2.4 Send as a text attachment (Plain Text)

Send As: Plain Text
To: schielke.stefan@gmail.com
From: <see testing >
Subject: Test
Body: This is the data to be sent as a text file

2.5 Send text attachment from phone directory path

Send As: HTML
To: schielke.stefan@gmail.com
From: Gmail account
Subject: Test
Body: This is the data to be sent as a text file
Attachment: Checked
Attachment Location: /accounts/1000/shared/documents/example.txt

2.6 Send multiple attachments from phone directory path

Send As: HTML
To: schielke.stefan@gmail.com
From: Gmail account
Subject: Test
Body: This is the data to be sent as a text file
Attachment: Checked
Attachment Location: /accounts/1000/shared/documents/example.txt,
/accounts/1000/shared/documents/example2.txt

2.7 send text attachment from phone sd card directory path

Send As: HTML
To: schielke.stefan@gmail.com
From: Gmail account
Subject: Test
Body: This is the data to be sent as a text file
Attachment: Checked
Attachment Location: /accounts/1000/removable/sdcard/For Testing/test.dat

2.8 Send multiple attachments from internal phone and SD Card paths

Send As: HTML
To: schielke.stefan@gmail.com
From: Gmail account
Subject: Test
Body: This is the data to be sent as a text file
Attachment: Checked
Attachment Location: /accounts/1000/shared/documents/example.txt,
/accounts/1000/removable/sdcard/For Testing/test.dat

Associated documentation – BlackBerry Developer – Cascades

Used for sending of attachments

Attachment
https://developer.blackberry.com/native/reference/cascades/bb__pim__message__attachment.html

QByteArray
https://developer.blackberry.com/native/reference/cascades/qbytearray.html

QString
https://developer.blackberry.com/native/reference/cascades/qstring.html

QUrl
https://developer.blackberry.com/native/reference/cascades/qurl.html

QFileInfo
https://developer.blackberry.com/native/reference/cascades/qfileinfo.html

Boost C++ Libraries

Welcome to Boost.org!

Boost provides free peer-reviewed portable C++ source libraries.

We emphasize libraries that work well with the C++ Standard Library. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use.

We aim to establish "existing practice" and provide reference implementations so that Boost libraries are suitable for eventual standardization. Ten Boost libraries are included in the C++ Standards Committee's Library Technical Report (TR1) and in the new C++11 Standard. C++11 also includes several more Boost libraries in addition to those from TR1. More Boost libraries are proposed for standardization in C++17. (Boost C++ Libraries)

EmailSender – Additional Functionality

Issues

Adding additional file paths to be accepted
Add functionality to strip Html tags for text messages
Add email signature
Add vCard

Details

Adding additional file paths to be accepted:
The last iteration only allowed for data to be pulled from the SD Card or the internal shared directories on the phone. Need to add the ability for different file paths as well as the local app sandbox

Add functionality to strip Html tags for text messages
When Plain Text is selected for the email format type, provide an option for the user to select to remove all of the html tags from the body of the email

Add email signature
Provide the ability to add an automatic email signature to the email

Add vCard
Provide the ability to add an automatic vCard to the email

Task – File Paths

Provide multiple file path options to be accepted

Steps taken

- Looked at the Android and iOS plugins to see what available options they used for file paths
- Researched what paths are available through BlackBerry
  - file:///... This is the full path of location on the device
  - /accounts/... This is just missing the file:// part of the path
  - ./... This access the app sandbox
  - /res/... This is the resource folder inside the app (requires ./app/native/)
- Added the checking of attachments and moved the code for checking the file path to its own function

emailSender_ndk.cpp

```
if (attachment.asString().compare("true") == 0){
   if (!attachmentLocation.empty()){
      foreach(Json::Value v, attachmentLocation){
         string checkpath = v.asString();
         QString path = checkPath(checkpath);
         attachFile(*builder, path);
      }
   }
   else{
      m_pParent->getLog()->info("No file path entered");
   }
}

...

QString EmailSenderNDK::checkPath(std::string checkpath){
   // remove any leading or trailing spaces from a file path
   char ws = ' ';
   checkpath = checkpath.erase(0, checkpath.find_first_not_of(ws));
   checkpath = checkpath.erase(checkpath.find_last_not_of(ws) + 1);

   QString forChecking = QString::fromStdString(checkpath);
   QString path;

   if(forChecking.startsWith(QString::fromStdString("file:///")) ||
         forChecking.startsWith(QString::fromStdString("./"))){
      path = forChecking;
   }
   else if(forChecking.startsWith(QString::fromStdString("/accounts/"))){
      path = QString::fromStdString("file://") + forChecking;
   }
   else if(forChecking.startsWith(QString::fromStdString("/res/"))){
      path = QString::fromStdString("./app/native/") + forChecking;;
   }
```

```
    else{
        path = forChecking;
    }
    return path;
}
```

- Moved the file attachment to its own function

emailSender_ndk.cpp

```
void EmailSenderNDK::attachFile(MessageBuilder& builder, QString path){
    QUrl filepath(path);
    QFileInfo fileinfo(path);

    if(!fileinfo.exists()){
        string nofile = "The file " + path.toStdString() + " cannot be found";
        char * cnofile = new char[nofile.length()+1];
        std::strcpy(cnofile, nofile.c_str());
        m_pParent->getLog()->info(cnofile);
    }
    else{
        QString filename = fileinfo.fileName();
        QString filetype = fileinfo.completeSuffix();
        Attachment msgAttach(filetype, filename, filepath);
        builder.addAttachment(msgAttach);
    }
}
```

- Test case 3.1
    - Results

      Gmail – Passed

Test Cases

3.1 Send from different path types

Send As: HTML
To: schielke.stefan@gmail.com
From: <see below>
Subject: Test Byte Array
Body: This a test message sending from multiple file locations
Attachment: Checked
Attachment Location: file:///accounts/1000/shared/documents/example.txt,
/accounts/1000/removable/sdcard/For Testing/test.dat,
./app/native/res/signature/example.vcf, /res/signature/example.txt

Task – Strip Html tags

Remove all of the html tags for a Plain Text message is the option is selected.

Steps taken

- Modified the index.html to add a checkbox for user option to remove tags

index.html

```
<label id='displayTxt'><input id="removeTags" type="checkbox" value="false">Remove html tags for plain text messages<br><br></label>
```

- Added the functionality to the index.js

index.js

```
if(document.getElementById('removeTags').checked)
   document.getElementById('removeTags').value = "true";
else
   document.getElementById('removeTags').value = "false";
var jsonEmail =
{
...
   "tags": document.getElementById('removeTags').value,
}
```

- Looked into the QtCore libraries to see if the functionality exists
- Nothing there so looked into common solutions for this issue
- Again, functionality available through Boost, or local coded solutions
- With some examples found through StackOverflow, I pieced together a solution and put it in its own function

emailSender_ndk.cpp

```
std::string EmailSenderNDK::stripHtml(std::string msgBody){
   std::vector<std::string> stripped;
   for(;;){
      std::string::size_type startpos;
      startpos = msgBody.find('<');
      if(startpos == std::string::npos){
         stripped.push_back(msgBody);
         break;
      }
      if(0 != startpos){
         stripped.push_back(msgBody.substr(0, startpos));
         msgBody = msgBody.substr(startpos, msgBody.size() - startpos);
         startpos = 0;
      }
      std::string::size_type endpos;
      for(endpos = startpos; endpos < msgBody.size() && msgBody[endpos] != '>'; ++endpos){
         if(msgBody[endpos] == '"'){
            endpos++;
            while(endpos < msgBody.size() && msgBody[endpos] != '"'){
               endpos++;
            }
```

```
            }
        }
        if(endpos == msgBody.size()){
            msgBody = msgBody.substr(endpos, msgBody.size() - endpos);
            break;
        }
        else{
            endpos++;
            msgBody = msgBody.substr(endpos, msgBody.size() - endpos);
        }
    }
    msgBody="";
    for(size_t i=0; i < stripped.size(); i++){
        msgBody += stripped[i];
    }
    return msgBody;
}
```

Test case 3.2

- Body of message

  This is a test message with *removal* of html tags from the message

- Results

  Gmail – Passed

Test case 3.3

- Body of message

  This is a test message with removal of html tags from the message

- Results

  Gmail – Passed

Test Cases

3.2 Send html message

Send As: Html
Remove Tags: un-checked
To: schielke.stefan@gmail.com
From: <see below>
Subject: Test Text with no tags
Body: This is a test message with removal <pre>of html tags from the message</pre>

3.3 Strip html tags for text messages

Send As: Plain Text
Remove Tags: Checked
To: schielke.stefan@gmail.com
From: <see below>
Subject: Test Text with no tags
Body: This is a test message with removal <pre>of html tags from the message</pre>

Task – Add Signature file

Add a signature file if user selects

Steps taken

- At first I just added a checkbox that the user could select that would automatically add a file that was in the app resource folder
- If the box was checked, the system would first look for a vCard file (example.vcf)
- If found, it would add that as the attachment
- If not, it would look for the signature file
- If the signature was found it would open the file, read the contents, and append them to the bottom of the message body
- If not found a info message would be sent to the logger
- I proposed this to Tim who said that he would prefer that these were their own functions and to allow the user to select the file location instead of a hardcoded path
- Additionally, he did not think that appending it to the message would be a useful function
- I reverted the files back and used the same functionality that the attachment would be using, so it was a simple addition to the files
- Modified the index.html to add a checkbox for user option to add signature file

index.html

```
<label><input id="signature" type="checkbox" value="false">Add signature</label><br>
<label id='displaySig'><input id="signatureLocation" type="text"
placeholder="/accounts/1000/.../file.txt"/><br><br></label>
```

- Added the functionality to the index.js

index.js

```
if(document.getElementById('signature').checked)
   document.getElementById('signature').value = "true";
else
   document.getElementById('signature').value = "false";
var jsonEmail =
{
...
   "signature": document.getElementById('signature').value,
}
```

- Added the code to the .cpp

emailSender_ndk.cpp

```
Json::Value signature = input["signature"];
Json::Value signatureLocation = input["signatureLocation"];
...
if (signature.asString().compare("true") == 0){
   if (!signatureLocation.empty()){
      foreach(Json::Value v, signatureLocation){
         string checkpath = v.asString();
         QString path = checkPath(checkpath);
         attachFile(*builder, path);
      }
   }
```

```
    else{
        m_pParent->getLog()->info("No file path entered");
    }
}
```

Test case 3.4

- Results
  Gmail – Passed

Test Cases

3.4 Add Signature Attachment

Send As: Html
To: schielke.stefan@gmail.com
From: <see below>
Subject: Test
Body: Testing signature attachment
Attachment: Checked
Attachment Location: /res/signature/signature.txt

Task – Add vCard file

Add a vCard file if user selects

Steps taken

- This functionality is identical to the signature function, so really just a duplicate (triplicate) of the attachment and signature functions
- Modified the index.html to add a checkbox for user option to add vCard file

index.html

```
<label><input id="vCard" type="checkbox" value="false">Add vCard</label><br>
<label id='displayVcard'><input id="vCardLocation" type="text"
placeholder="/accounts/1000/.../file.vcf"/><br><br></label>
```

- Added the functionality to the index.js

index.js

```
if(document.getElementById('vCard').checked)
   document.getElementById('vCard').value = "true";
else
   document.getElementById('vCard').value = "false";
var jsonEmail =
{
...
   "vCard": document.getElementById('vCard').value,
}
```

- Added the code to the .cpp

emailSender_ndk.cpp

```
if (vCard.asString().compare("true") == 0){
   if (!vCardLocation.empty()){
      foreach(Json::Value v, vCardLocation){
         string checkpath = v.asString();
         QString path = checkPath(checkpath);
         attachFile(*builder, path);
      }
   }
   else{
      m_pParent->getLog()->info("No file path entered");
   }
}
```

Test case 3.5

- Results

  Gmail – Passed

Test Cases

3.5 Add vCard Attachment

Send As: Html
To: schielke.stefan@gmail.com
From: <see below>
Subject: Test
Body: Testing vCard attachment
Attachment: Checked
Attachment Location: ./app/native/res/signature/vCard.vcf

Lessons Learned, UCOSP Experience & Wrap-up

Lessons Learned

- Great experience learning from code reviews
- Development in real-world production code base
- Real-world experience is very similar to University assignments in that nothing goes easy
- Using someone else's base code and developing and enhancing theirs can be an challenge
- Lots of documentation to read, understand, and utilize (Cascades & Core)
- Learned how to use Git and GitHub which I found to be quite a valuable tool that I will surely use
- Gained practical skills and experience that I can take out into my career
- Proved that TRU students can work with the best Universities in Canada and compete at the same level

UCOSP Experience

I was expecting more from this course to be honest. All of us in our team basically worked on individual tasks. There was not the collaboration and experience of sharing code that I would have liked to have gotten from this experience.

For example, in the project description it states, "Students should be familiar with C++ or JavaScript development, though they do not need to be experts with both. We will work in pairs and teams to round out the necessary skill set." (UCOSP - Projects). I definitely was more comfortable with C++ over JavaScript going into the course. Most all questions and problems that I experienced I had to discover solutions on my own. It would have been nice to have someone that had the experience in the other areas in order to save some time on researching what were sometimes simple solutions, but a lot of research.

Additionally, in the about section of the UCOSP website it states, "What level of work is expected? (Almost) all of these projects are producing software for real-world use, so standards are high. Remember, "95% correct" may be an 'A' academically, but if 5% of an application is buggy, users aren't going to be happy." (UCOSP - About). I felt that I could have easily coded solutions that would have met the requirements of the task, but, I did not feel that providing solutions that worked "most" of the time was the best use of this experience or what I was here to do. I spent a lot or time trying to fully understand what the libraries available had to use to ensure that the code that I provided was the 'best' solution that I was able to offer.

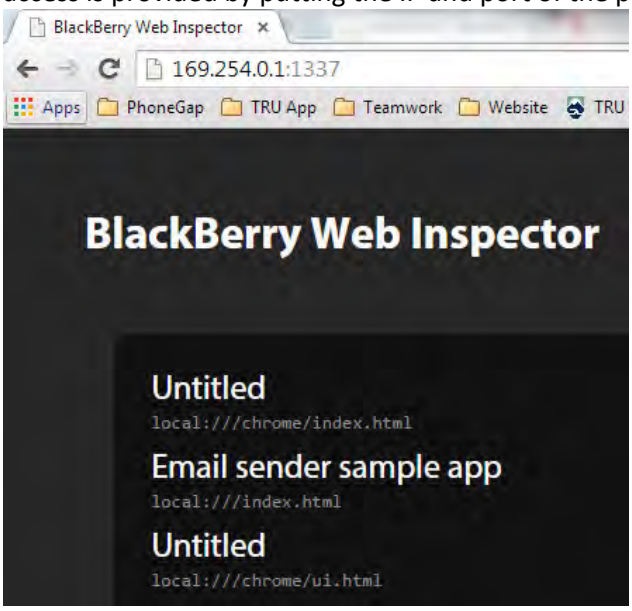
Wrap-up

While I did experience frustrations along the way I was generally pleased with the overall experience. I was in high spirits through the whole process and feel that I have learned a lot and can use this experience going forward in both my personal projects as well as future employment.

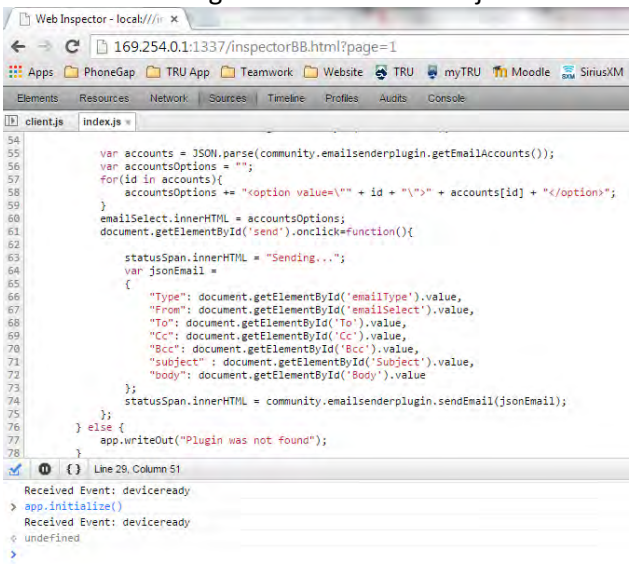
I have mentioned to Tim that I would be interested in working on some projects in the future should anything arise that he feels I could help with. He was open to this, so I hope that something can come out of it. But overall, it was a great opportunity that I am truly grateful that I was able to participate in. Thank you Kevin and TRU for this once in a lifetime University experience.

Debugging

While we were at the code sprint we were shown how to use the 'BlackBerry Web Inspector'. This access is provided by putting the IP and port of the phone into a browser.



Clicking on the Email sender sample app would provide us with the access to the source files (.js) as well as console messages from alerts in the .js files.



This was useful for making sure that our calls were being accessed, but it would still not show us any problems or any debugging in the .cpp files. And this is generally where we needed to see what was happening with our files as this is where all of the magic really happened.

On September 29th we received an email from Tim that provided us with instructions on using the 'Logger' that is part of most of the plugins. This definitely helped us bridge the gap and enabled us to be able to see what was happening in our files.

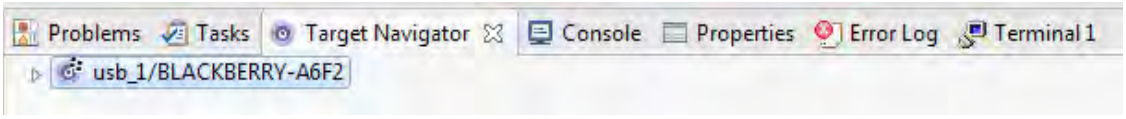
"The Logger class wraps the slog2 API offered by the BlackBerry 10 native SDK. It is instantiated as part of the JS portion of the plugin, and typically accessible by calling getLog() on the m_pParent pointer within the NDK portion of a plugin (where you write the bulk of the native code). It's main goal is just to make it simple to write out data in string format to the log, within your code, so you can see what is going on, but also so that basic warning and debugging information is included in the plugin. For that reason it includes several different types of log level, and maps them to the various severities that the slog2 API supports.

To view the log, you can attach Momentics to your device and view the output. In the main Momentics menu, go to Window->Show View, and select Target Navigator. That view should appear in the lower panel. If there's no target listed for your device, right click in the panel and select New BlackBerry Target. Fill out the fields for your device, and click okay. When you've got a device listed there, right click on it and select Connect if you aren't already connected. To view the log, right click on the device target and choose Open Device Log. That will show you what's being output by the system. It's filtered to just content from apps running in debug mode, so there won't be a lot besides what you output yourself.

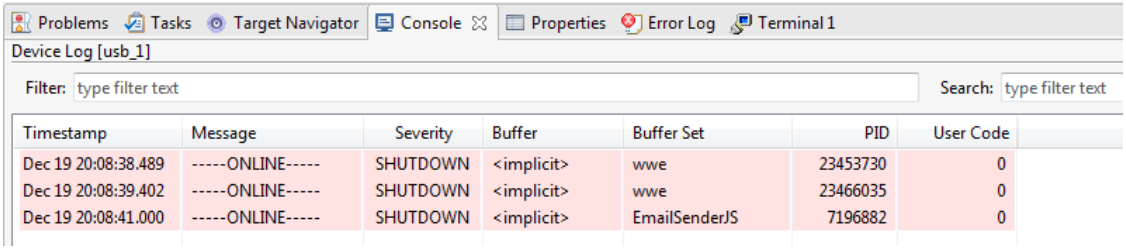
Alternatively, you can use the Launch SSH Session command on the target instead. This connects you over SSH to the device so you can write typical POSIX commands in a terminal. The log can be viewed by running slog2info. Adding a –w flag will make the output wait and output new content as it arrives. I prefer the log output the other way, but this is useful for exploring the filesystem if you need it." (Windsor, Using the Logger and other plugins, 2014)

Steps in Momentics

Enabling access to view output on the device in the 'Target Navigator' tab



Where any output messages from the .cpp or .js files would appear in the 'Console' tab.



To output a message from the .cpp files, an example of a statement could be used:

```
m_pParent->getLog()->info("No file path entered");
```

There are different levels of messages that can be used depending on the intent of the message.

From the Logger.hpp file:

int **debug**(const char* message);
int **info**(const char* message);
int **notice**(const char* message);
int **warn**(const char* message);
int **error**(const char* message);
int **critical**(const char* message);

Appendix

Weekly Documentation

Week 1: Weekly Report for COMP 4480 - UCOSP Directed Studies

BlackBerry / PhoneGap
Mentor: Tim Windsor
Monday, September 10th 2014

Summary:

During the first week, I received instructions from my mentor, Tim Windsor of BlackBerry, for installing the necessary software onto our own environments. I spoke with Gavin about his experiences from the winter semester, and he gave Tim and I an introduction to Git. Additionally we had our first group meeting through Google Hangouts.

Task Completion from First Week

- First contact from BlackBerry mentor Tim Windsor with instructions for setting up our work environments – Tuesday, Sept 2
- Installed onto my machine - Tuesday, Sept 9
    - WebWorks SDK
    - Momentics NDK
- Had an introduction to Git by Gavin – Wednesday, Sept 10
- Forked to Blackberry Git main repo – Wednesday, Sept 10
    - Installed Git onto local machine
    - Imported the projects into Momentics
- First meeting with UCOSP team members (Google+ Hangout) – Saturday, Sept 14
    - Introduction to all team members >
    - Introduction to tasks by Tim Windsor
    - Informed that we will all receive a BlackBerry Z10 or Q10 for development work on the project

Task for This Week

- Fly to Toronto on Thursday, Sept 18 for Code Sprint
    - Discuss with Tim, what his expectations are
    - Learn how to create an API for BlackBerry
- Review the tasks and become familiar with what each one is and should do

Additional Information

- None this week

Known issues / things blocking progress

- New environment to become familiar with (although very similar to Eclipse)
- Need to understand and learn Git better

Weeks 2-3: Weekly Report for COMP 4480 - UCOSP Directed Studies

BlackBerry / PhoneGap
Mentor: Tim Windsor
Monday, September 29th 2014

Summary:

Our second week consisted of the code sprint in Toronto. The following week was more catch-up on other courses.

Task Completion from Weeks 2-3

- Environment set-up complete
- Tasks chosen at the code sprint.
    - emailSender
- Becoming more familiar with the development and processes

Task for This Week

- Get the phone accounts to be recognized by the program
- Test HTML vs plaintext message sending / receiving

Additional Information

- After our weekly conference call on Friday, Tim said he would send us instructions on using the logging and debugging tools
- Working with Tim Tung to discuss issues between ourselves and try to help each other out with becoming familiar.

Known issues / things blocking progress

- New environment to become familiar with
- Need to understand and learn Git better
- Debugging issues between runtime WebWorks and the C++ ndk

Weeks 4: Weekly Report for COMP 4480 - UCOSP Directed Studies

BlackBerry / PhoneGap
Mentor: Tim Windsor
Monday, October 6th 2014

Summary:

I was finally able to get the plugin to recognize the accounts on the phone. The original config.xml that is in the sample of the emailSender plugin includes the permissions:

```
<rim:permissions>
   <rim:permit>access_pimdomain_contacts</rim:permit>
   <rim:permit>access_pimdomain_messages</rim:permit>
</rim:permissions>
```

But when executing the files on my end, the config.xml file did not include the permissions. Once these permissions were added, the phone had no problems pulling up the accounts.
I have three different accounts on the device: gmail, yahoo, and outlook. All three accounts pull up in the plugin correctly. The Readme.md from the plugin says that some of the account types will not work (Yahoo!, Microsoft accounts – Hotmail/Outlook). I can confirm that the Outlook account does not send, but Gmail & Yahoo! work fine.

Task Completion from Week 4

- emailSender is working and pulling the account info from the phone
- testing of sending emails from all accounts
- testing different email formats (html, plain text)

Task for This Week

- finish testing of email formats
- Start looking into the attachment types (second task)

Additional Information

- Working with Tim Tung to discuss issues between ourselves and try to help each other out on understanding tasks & system.
- The whole process is going slower than I thought it would. While I find this frustrating, I am trying to keep my spirits up! ☺

Known issues / things blocking progress

- New environment to become familiar with
- Need to understand and learn Git better
- Debugging issues between runtime WebWorks and the C++ ndk – still having some issues

Weeks 5-6: Weekly Report for COMP 4480 - UCOSP Directed Studies

BlackBerry / PhoneGap
Mentor: Tim Windsor
Monday, October 20th 2014

Summary:

I ran into an issue where the build was showing up in the plugin, but not reflected on the phone. After a lot of trial and error, I was able to get Tim Windsor to take a look on my machine. After ½ hour of checking configurations and settings it was found that the build was only taking place for the 'device-simulator' and 'simulator'. It was not actually building for the 'device'. After this change was made, it had no issues with processing on the phone. But when the program is shut down, it defaults back to the 'device-simulator' and 'simulator' and has to be changed when restarted.
I ran multiple tests from all accounts, using both the html and plain text formats. All files were updated to with the current changes and a pull request was made. It has been merged into the BB10-Cordova library now.
I have started looking into the documentation for what is required for sending attachments. I have some ideas for how to proceed and will start the coding this Wednesday.

Task Completion from Weeks 5-6

- emailSender is working and pulling the account info from the phone
- testing of sending emails from all accounts
- testing different email formats (html, plain text)
- pull-request made
- code merged
- documentation reading on email attachments

Task for This Week

- coding for adding attachment function to the plugin

Additional Information

- Working with Tim Tung to discuss issues between ourselves and try to help each other out on understanding tasks & system.
- The whole process is going slower than I thought it would. While I find this frustrating, I am trying to keep my spirits up! ☺

Known issues / things blocking progress

- New environment to become familiar with
- Need to understand and learn Git better
- Debugging issues between runtime WebWorks and the C++ ndk – still having some issues
- Issues that arise with the IDE that are not clear on what is been done and why

Week 7: Weekly Report for COMP 4480 - UCOSP Directed Studies

BlackBerry / PhoneGap
Mentor: Tim Windsor
Monday, October 27th 2014

Summary:

I have added the email attachment functionality to the plugin
The addition to the plugin asks for:
MessageBuilder can also take Attachment(s) that can be constructed from strings (or byte arrays):
https://developer.blackberry.com/native/reference/cascades/bb__pim__message__attachment.html#function-attachment-mimetype-name-textdata
Supporting attachments in this way would be beneficial because it would allow webworks apps to append attachments to emails without requiring 'Shared File' permissions (currently the invoke card for email requires a path to the shared directory for attachments; private app storage cannot be used).
What is unclear is what data the plugin should be able to access? The app and shared files, or '*other*' apps data?

Task Completion from Week 7

- emailSender is working and pulling the files from the phone
- testing of sending emails from all a**ccounts (gmail, yahoo)**
- testing different email attachments (txt, docx, dat, qArray)
- looked up additional documentation on email attachments

Task for This Week

- finish coding for adding attachment function to the plugin

Additional Information

- Waiting for reply from Tim Windsor as to what exactly the plugin should be doing

Known issues / things blocking progress

- New environment to become familiar with
- Need to understand and learn Git better
- Issues that arise with the IDE that are not clear on what is been done and why

Week 8: Weekly Report for COMP 4480 - UCOSP Directed Studies

BlackBerry / PhoneGap
Mentor: Tim Windsor
Monday, November 3rd 2014

Summary:

I have added the email attachment functionality to the plugin
I am able to send any file format from the shared folders on the phone. I do not have a SD card to test from. I will purchase on and make sure that the plugin can access the data off of the card

The reply from Tim and discussion of Friday cleared up what the plugin should access, but now there is an issue with sending multiple files from the plugin.

As per Tim, there is not a lot left in his tasks that he would like to get done.
He has asked if there is anything that we would like to do that might be cool.

Task Completion from Week 8

- emailSender is working and pulling the files from the phone
- testing different email attachments (txt, docx, dat, qArray)
- looked up additional documentation on email attachments

Task for This Week

- Continue testing of sending multiple attachments
- Purchase and test sending from SD card
- Add FilePicker functionality so that the user can traverse storage and pick their file(s)

Additional Information

- Waiting for reply from Tim Windsor as to what format he would like me to proceed with

Known issues / things blocking progress

- Need to understand and learn Git better

Week 9: Weekly Report for COMP 4480 - UCOSP Directed Studies

BlackBerry / PhoneGap
Mentor: Tim Windsor
Monday, November 10th 2014

Summary:

Basic functionality has been tested and pull request sent.
Working on additional file paths and error checking for the plugin.
Asked if the addition of an automatic email signature would be beneficial. Tim said to add it so I am looking at ways to include it in the messages.

Task Completion from Week 9

- Purchased micro sd card and testing attachment pulling
- Submitted Pull Request for attachment functionality

Task for This Week

- Add attachment functionality from additional paths (local app, additional paths, html?)
- Look at adding a .txt file for email signature that could be added automatically to an attachment
- Update blog for sending to Tim.

Additional Information

- Updating reports and project details

Known issues / things blocking progress

- None

Week 10: Weekly Report for COMP 4480 - UCOSP Directed Studies

BlackBerry / PhoneGap
Mentor: Tim Windsor
Monday, November 17th 2014

Summary:

Added functionality for the user to request an automatic vCard or email signature
During test I have found that there is an issues with the way that the data is put into the email for html emails. I have asked Tim if he would like a bug report to be created or if I should research the issue and try and resolve the problem.

Task Completion from Week 10

- emailSender adds the signature file based off user request.
    - the user must add a .vcf or .txt file into the \res\signature folder of the app

Task for This Week

- Update blog for sending to Tim
- Requested another task (Add BlackBerry 10 support to App Preferences API #341)

Additional Information

- Waiting for reply from Tim Windsor as to what he would like me to proceed with.

Known issues / things blocking progress

- None

Week 11: Weekly Report for COMP 4480 - UCOSP Directed Studies

BlackBerry / PhoneGap
Mentor: Tim Windsor
Monday, November 24th 2014

Summary:

Changed the way that the paths pull the data from the plugin. User to specify the path and not have it hard-coded. Turns out I've spent some unnecessary time on formatting the sample plugin. As long as the ndk works the plugin is not important!

Task Completion from Week 11

- changed format for where data is pulled from

Task for This Week

- Finalize that format is acceptable

Additional Information

- None

Known issues / things blocking progress

- None

Code from the start of the project

index.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="format-detection" content="telephone=no" />
    <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1, width=device-width, height=device-height, target-densitydpi=device-dpi" />
    <link rel="stylesheet" type="text/css" href="css/index.css" />
    <title>Email sender sample app</title>
  </head>
  <body>
    <div class="app">
      <h1>Apache Cordova</h1>
      <div id="deviceready" class="blink">
        <p class="event listening">Connecting to Device</p>
        <p class="event received">Device is Ready</p>
      </div>
      <div id="results">
        <h3>Email sender sample app</h3>
        <label>Send from:</label><br><select id="emailSelect"></select>
        <label>To:</label><br><input id="To" type="text" placeholder="To"/><br><br>
        <label>Cc:</label><br><input id="Cc" type="text" placeholder="Cc"/><br><br>
        <label>Bcc:</label><br><input id="Bcc" type="text" placeholder="Bcc"/><br><br>
        <label>Subject:</label><br><input id="Subject" type="text" placeholder="Subject"/><br><br>
        <label>Body:</label><br><textarea id="Body" placeholder="Body" rows="5"></textarea><br>
        <input id="send" type="button" value="send"/>
        <p>Status: <span id="status"> - </span></p>
      </div>
    </div>
    <script type="text/javascript" src="cordova.js"></script>
    <script type="text/javascript" src="js/index.js"></script>
    <script type="text/javascript">
      app.initialize();
    </script>
  </body>
</html>
```

index.js

```
testPluginCalls: function() {

…
        statusSpan.innerHTML = "Sending...";
        var jsonEmail =
        {
          "From": document.getElementById('emailSelect').value,
```

```
            "To": document.getElementById('To').value,
            "Cc": document.getElementById('Cc').value,
            "Bcc": document.getElementById('Bcc').value,
            "subject" : document.getElementById('Subject').value,
            "body": document.getElementById('Body').value,
        };
        statusSpan.innerHTML = community.emailsenderplugin.sendEmail(jsonEmail);
…
}
```

emailSender_ndk.cpp

```
std::string EmailSenderNDK::sendEmail(const std::string& inputString) {
   Json::FastWriter writer;
   Json::Reader reader;
   Json::Value input;
   bool parse = reader.parse(inputString, input);
   if(parse){
      Json::Value From = input["From"];
      Json::Value To = input["To"];
      Json::Value Cc = input["Cc"];
      Json::Value Bcc = input["Bcc"];
      Json::Value subject = input["subject"];
      Json::Value body = input["body"];

      long id = atol(From.asString().c_str());
      Account account;

      if(id == -1){
         account = accountService.defaultAccount(Service::Messages);
      }
      else{
         account = accountService.account(id);
      }
      if(!account.isValid()) return "The account is not valid.";
      MessageBuilder *builder = MessageBuilder::create(account.id());

      if(To.isArray()){
         foreach(Json::Value v, To){
            QString email =  QString::fromStdString(v.asString());
            MessageContact rto = MessageContact(-1,MessageContact::To,  email, email);
            builder->addRecipient(rto);
         }
      }
      else{
         QString email =  QString::fromStdString(To.asString());
         MessageContact rto = MessageContact(-1,MessageContact::To, email, email);
         builder->addRecipient(rto);
      }
```

```
    if(Cc.isArray()){
      foreach(Json::Value v, Cc){
        QString email =  QString::fromStdString(v.asString());
        MessageContact rCc = MessageContact(-1,MessageContact::Cc, email, email);
        builder->addRecipient(rCc);
      }
    }
    else{
      QString email =  QString::fromStdString(Cc.asString());
      MessageContact rCc = MessageContact(-1,MessageContact::Cc, email, email);
      builder->addRecipient(rCc);
    }

    if(Bcc.isArray()){
      foreach(Json::Value v, Bcc){
        QString email =  QString::fromStdString(v.asString());
        MessageContact rBcc = MessageContact(-1,MessageContact::Bcc, email, email);
        builder->addRecipient(rBcc);
      }
    }
    else{
      QString email =  QString::fromStdString(Bcc.asString());
      MessageContact rBcc = MessageContact(-1,MessageContact::Bcc, email, email);
      builder->addRecipient(rBcc);
    }

    builder->subject(QString::fromStdString(subject.asString()));
    string stringBody = body.asString();
    QByteArray bodyData ;
    bodyData.append(QString::fromStdString(body.asString()));
    builder->body(MessageBody::PlainText,bodyData);
    Message m = *builder;
    MessageKey mk = messageService.send (m.accountId(), m );

    if(mk == 0){
      return "The email has not been sent.";
    }
    else{
      return "Sent.";
    }
  }
  else{
    return "The JSON can't be parse.";
  }

  return "An error as occurred.";
}
```

Code from the end of the project

index.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="format-detection" content="telephone=no" />
    <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1, width=device-width, height=device-height, target-densitydpi=device-dpi" />
    <link rel="stylesheet" type="text/css" href="css/index.css" />
    <title>Email sender sample app</title>
  </head>
  <body>
    <div class="app">
      <h1>Apache Cordova</h1>
      <div id="deviceready" class="blink">
        <p class="event listening">Connecting to Device</p>
        <p class="event received">Device is Ready</p>
      </div>
      <div id="results">
        <h3>Email sender sample app</h3>
        <label>Send as:</label><br><select id="emailType"><option value="html">HTML</option><option value="txt">Plain Text</option></select><br><br>
         <label id='toDisplay'><input id="removeTags" type="checkbox" value="false">Remove html tags for plain text messages<br><br></label>
        <label>Send from:</label><br><select id="emailSelect"></select><br><br>
        <label>To:</label><br><input id="To" type="text" placeholder="To"/><br><br>
        <label>Cc:</label><br><input id="Cc" type="text" placeholder="Cc"/><br><br>
        <label>Bcc:</label><br><input id="Bcc" type="text" placeholder="Bcc"/><br><br>
        <label>Subject:</label><br><input id="Subject" type="text" placeholder="Subject"/><br><br>
         <label>Body:</label><br><textarea id="Body" placeholder="Body" rows="5"></textarea><br><br>
        <label><input id="vCard" type="checkbox" value="false">Add vCard</label><br>
        <input id="vCardLocation" type="text" placeholder="/accounts/1000/.../file.vcf" rows="2"/><br><br>
        <label><input id="signature" type="checkbox" value="false">Add signature</label><br>
        <input id="signatureLocation" type="text" placeholder="/accounts/1000/.../file.txt" rows="2"/><br><br>
        <label><input id="attachment" type="checkbox" value="false">Add attachment</label><br>
        <input id="attachmentLocation" type="text" placeholder="/accounts/1000/.../file.ext" rows="2"/><br><br>
        <input id="send" type="button" value="send"/>
        <p>Status: <span id="status"> - </span></p>
      </div>
    </div>
    <script type="text/javascript" src="cordova.js"></script>
    <script type="text/javascript" src="js/index.js"></script>
```

```
    <script type="text/javascript">
      app.initialize();
    </script>
  </body>
</html>
```

index.js

```
testPluginCalls: function() {
…
        statusSpan.innerHTML = "Sending...";
        if(document.getElementById('removeTags').checked)
           document.getElementById('removeTags').value = "true";
        else
           document.getElementById('removeTags').value = "false";
        if(document.getElementById('vCard').checked)
           document.getElementById('vCard').value = "true";
        else
           document.getElementById('vCard').value = "false";
        if(document.getElementById('signature').checked)
           document.getElementById('signature').value = "true";
        else
           document.getElementById('signature').value = "false";
        if(document.getElementById('attachment').checked)
           document.getElementById('attachment').value = "true";
        else
           document.getElementById('attachment').value = "false";

        var jsonEmail =
        {
           "Type": document.getElementById('emailType').value,
           "tags": document.getElementById('removeTags').value,
           "From": document.getElementById('emailSelect').value,
           "To": document.getElementById('To').value,
           "Cc": document.getElementById('Cc').value,
           "Bcc": document.getElementById('Bcc').value,
           "subject" : document.getElementById('Subject').value,
           "body": "<pre>" + document.getElementById('Body').value + "</pre>",
           "vCard": document.getElementById('vCard').value,
           "vCardLocation": document.getElementById('vCardLocation').value.split(','),
           "signature": document.getElementById('signature').value,
           "signatureLocation": document.getElementById('signatureLocation').value.split(','),
           "attachment": document.getElementById('attachment').value,
           "attachmentLocation": document.getElementById('attachmentLocation').value.split(',')
        };
        statusSpan.innerHTML = community.emailsenderplugin.sendEmail(jsonEmail);

…
}
```

emailSender_ndk.cpp

```
std::string EmailSenderNDK::sendEmail(const std::string& inputString) {
    Json::FastWriter writer;
    Json::Reader reader;
    Json::Value input;
    bool parse = reader.parse(inputString, input);
    if(parse){
        Json::Value Type = input["Type"];
        Json::Value tags = input["tags"];
        Json::Value From = input["From"];
        Json::Value To = input["To"];
        Json::Value Cc = input["Cc"];
        Json::Value Bcc = input["Bcc"];
        Json::Value subject = input["subject"];
        Json::Value body = input["body"];
        Json::Value vCard = input["vCard"];
        Json::Value vCardLocation = input["vCardLocation"];
        Json::Value signature = input["signature"];
        Json::Value signatureLocation = input["signatureLocation"];
        Json::Value attachment = input["attachment"];
        Json::Value attachmentLocation = input["attachmentLocation"];

        long id = atol(From.asString().c_str());
        Account account;

        if(id == -1){
            account = accountService.defaultAccount(Service::Messages);
        }
        else{
            account = accountService.account(id);
        }
        if(!account.isValid()) return "The account is not valid.";
        MessageBuilder *builder = MessageBuilder::create(account.id());

...

        builder->subject(QString::fromStdString(subject.asString()));

        std::string msgBody = body.asString();

        if(Type != "html" && tags.asString().compare("true") == 0){
            msgBody = stripHtml(msgBody); // strip tags for text messages
        }

        QByteArray bodyData;
        bodyData.append(QString::fromStdString(msgBody));
        if(Type=="html"){
            builder->body(MessageBody::Html, bodyData);
```

```
    }
    else{
        builder->body(MessageBody::PlainText, bodyData);
    }

    if (vCard.asString().compare("true") == 0){
        if (!vCardLocation.empty()){
            foreach(Json::Value v, vCardLocation){
                string checkpath = v.asString();
                QString path = checkPath(checkpath);
                attachFile(*builder, path);
            }
        }
        else{
            m_pParent->getLog()->info("No file path entered");
        }
    }

    if (signature.asString().compare("true") == 0){
        if (!signatureLocation.empty()){
            foreach(Json::Value v, signatureLocation){
                string checkpath = v.asString();
                QString path = checkPath(checkpath);
                attachFile(*builder, path);
            }
        }
        else{
            m_pParent->getLog()->info("No file path entered");
        }
    }

    if (attachment.asString().compare("true") == 0){
        if (!attachmentLocation.empty()){
            foreach(Json::Value v, attachmentLocation){
                string checkpath = v.asString();
                QString path = checkPath(checkpath);
                attachFile(*builder, path);
            }
        }
        else{
            m_pParent->getLog()->info("No file path entered");
        }
    }

    Message m = *builder;
    MessageKey mk = messageService.send (m.accountId(), m );

    if(mk == 0){
```

```
            return "The email has not been sent.";
        }
        else{
            return "Sent.";
        }
    }
    else{
        return "The JSON can't be parsed.";
    }
    return "An error as occurred.";
}

std::string EmailSenderNDK::stripHtml(std::string msgBody){
    std::vector<std::string> stripped;
    for(;;){
        std::string::size_type startpos;
        startpos = msgBody.find('<');
        if(startpos == std::string::npos){
            stripped.push_back(msgBody);
            break;
        }
        if(0 != startpos){
            stripped.push_back(msgBody.substr(0, startpos));
            msgBody = msgBody.substr(startpos, msgBody.size() - startpos);
            startpos = 0;
        }
        std::string::size_type endpos;
        for(endpos = startpos; endpos < msgBody.size() && msgBody[endpos] != '>'; ++endpos){
            if(msgBody[endpos] == '"'){
                endpos++;
                while(endpos < msgBody.size() && msgBody[endpos] != '"'){
                    endpos++;
                }
            }
        }
        if(endpos == msgBody.size()){
            msgBody = msgBody.substr(endpos, msgBody.size() - endpos);
            break;
        }
        else{
            endpos++;
            msgBody = msgBody.substr(endpos, msgBody.size() - endpos);
        }
    }
    msgBody="";
    for(size_t i=0; i < stripped.size(); i++){
        msgBody += stripped[i];
    }
```

```
    return msgBody;
}

QString EmailSenderNDK::checkPath(std::string checkpath){
    // remove any leading or trailing spaces from a file path
    char ws = ' ';
    checkpath = checkpath.erase(0, checkpath.find_first_not_of(ws));
    checkpath = checkpath.erase(checkpath.find_last_not_of(ws) + 1);

    // allow for full or partial path to be entered
    QString forChecking = QString::fromStdString(checkpath);
    QString path;

    if(forChecking.startsWith(QString::fromStdString("file:///")) ||
forChecking.startsWith(QString::fromStdString("./"))){
        path = forChecking;
    }
    else if(forChecking.startsWith(QString::fromStdString("/accounts/"))){
        path = QString::fromStdString("file://") + forChecking;
    }
    else if(forChecking.startsWith(QString::fromStdString("/res/"))){
        path = QString::fromStdString("./app/native/") + forChecking;;
    }
    else{
        path = forChecking;
    }
    return path;
}

void EmailSenderNDK::attachFile(MessageBuilder& builder, QString path){
    QUrl filepath(path);
    QFileInfo fileinfo(path);

    if(!fileinfo.exists()){
        string nofile = "The file " + path.toStdString() + " cannot be found";
        char * cnofile = new char[nofile.length()+1];
        std::strcpy(cnofile, nofile.c_str());
        m_pParent->getLog()->info(cnofile);
    }
    else{
        QString filename = fileinfo.fileName();
        QString filetype = fileinfo.completeSuffix();
        Attachment msgAttach(filetype, filename, filepath);
        builder.addAttachment(msgAttach);
    }
}
```

Bibliography

Boost C++ Libraries. (n.d.). Retrieved December 18, 2014, from Boost C++ Libraries: http://www.boost.org/

Git (software). (n.d.). Retrieved December 18, 2014, from Wikipedia: https://en.wikipedia.org/wiki/Git_(software)

GitHub. (n.d.). Retrieved December 18, 2014, from Wikipedia: https://en.wikipedia.org/wiki/GitHub

Notepad++. (n.d.). Retrieved December 18, 2014, from About: http://notepad-plus-plus.org/

QNX Momentics Tool Suite. (n.d.). Retrieved December 18, 2014, from QNX: http://www.qnx.com/products/tools/qnx-momentics.html

Tim Windsor. (n.d.). Retrieved December 18, 2014, from LinkedIn: https://www.linkedin.com/in/timwindsor

UCOSP - About. (n.d.). Retrieved December 18, 2014, from UCOSP: http://ucosp.ca/about/

UCOSP - Projects. (n.d.). Retrieved December 18, 2014, from UCOSP: http://ucosp.ca/projects/

Windsor, T. (2014, November 5). RE: Update to emailSender. ON, Canada.

Windsor, T. (2014, September 29). Using the Logger and other plugins. ON, Canada.