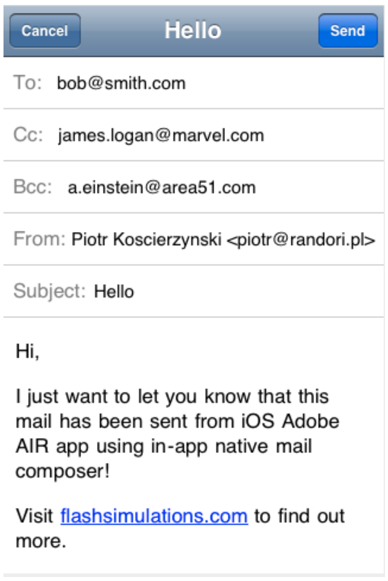
Cordova EmailComposer Plugin



To: bob@smith.com

Cc: james.logan@marvel.com

Bcc: a.einstein@area51.com

From: Piotr Koscierzynski <piotr@randori.pl>

Subject: Hello

Hi,

I just want to let you know that this mail has been sent from iOS Adobe AIR app using in-app native mail composer!

Visit flashsimulations.com to find out more.

The plugin provides access to the standard interface that manages the editing and sending an email message. You can use this view controller to display a standard email view inside your application and populate the fields of that view with initial values, such as the subject, email recipients, body text, and attachments. The user can edit the initial contents you specify and choose to send the email or cancel the operation.

Using this interface does not guarantee immediate delivery of the corresponding email message. The user may cancel the creation of the message, and if the user does choose to send the message, the message is only queued in the Mail application outbox. This allows you to generate emails even in situations where the user does not have network access, such as in airplane mode. This interface does not provide a way for you to verify whether emails were actually sent.

Plugin's Purpose

The purpose of the plugin is to create an platform independent javascript interface for Cordova based mobile applications to access the specific email composition API on each platform.

Overview

Supported Platforms

- **iOS** *(up to iOS8)*
- **Android** *(up to KitKat and L)*
- **WP8** *(up to WP8.1)*

Installation

The plugin can either be installed from git repository, from local file system through the Command-line Interface. Or cloud based through PhoneGap Build.

Local development environment

From master:

```
# ~~ from master branch ~~
cordova plugin add https://github.com/katzer/cordova-plugin-email-composer.git
```

from a local folder:

```
# ~~ local folder ~~
cordova plugin add de.appplant.cordova.plugin.email-composer --searchpath path/to/plugin
```

or to use the last stable version:

```
# ~~ stable version ~~
cordova plugin add de.appplant.cordova.plugin.email-composer@0.8.2
```

PhoneGap Build

Add the following xml to your config.xml to always use the latest version of this plugin:

```
<gap:plugin name="de.appplant.cordova.plugin.email-composer" version="0.8.2" />
```

More informations can be found here.

ChangeLog

Version 0.8.2 (not yet released)

- Added new namespace `cordova.plugins.email`
  Note: The former `plugin.email` namespace is now deprecated and will be removed with the next major release.
- [***change:***] Unified `absolute:` and `relative:` to `file:`
- [***change:***] Renamed `isServiceAvailable` to `isAvailable`
- [feature:] `res:` prefix for native ressource attachments
- [enhancement:] `open` supports callbacks
- [enhancement:] `isHTML` can be used next `isHtml`
- [bugfix:] Defaults were ignored

Further informations

- See CHANGELOG.md to get the full changelog for the plugin.

Using the plugin

The plugin creates the object `cordova.plugins.email` with following methods:

Plugin initialization

The plugin and its methods are not available before the *deviceready* event has been fired.

```
document.addEventListener('deviceready', function () {
    // cordova.plugins.email is now available
}, false);
```

Determine if the device is capable to send emails

The ability to send emails can be revised through the `email.isAvailable` interface. The method takes a callback function, passed to which is a boolean property. Optionally the callback scope can be assigned as a second parameter.

The Email service is only available on devices capable which are able to send emails. E.g. which have configured an email account and have installed an email app. You can use this function to hide email functionality from users who will be unable to use it.

```
cordova.plugins.email.isAvailable(
    function (isAvailable) {
        // alert('Service is not available') unless isAvailable;
    }
);
```

Open a pre-filled email draft

A pre-filled email draft can be opened through the `email.open` or `email.openDraft` interface. The method takes a hash as an argument to specify the email's properties. All properties are optional. Further more it accepts an callback function to be called after the email view has been dismissed.

After opening the draft the user may have the possibilities to edit, delete or send the email.

Further informations

- An configured email account is required to send emails.
- Attachments can be either base64 encoded datas, files from the the device storage or assets from within the *www* folder.
- The default value for *isHTML* is *true*.
- See the examples for how to create and show an email draft.

```
cordova.plugins.email.open({
    to:          Array, // email addresses for TO field
    cc:          Array, // email addresses for CC field
    bcc:         Array, // email addresses for BCC field
    attachments: Array, // file paths or base64 data streams
    subject:    String, // subject of the email
    body:       String, // email body (for HTML, set isHtml to true)
    isHtml:    Boolean, // indicats if the body is HTML or plain text
}, callback, scope);
```

Examples

Open an email draft

The following example shows how to create and show an email draft pre-filled with different kind of properties.

```
cordova.plugins.email.open({
    to:      'max@mustermann.de',
    cc:      'erika@mustermann.de',
    bcc:     ['john@doe.com', 'jane@doe.com'],
    subject: 'Greetings',
    body:    'How are you? Nice greetings from Leipzig'
});
```

Of course its also possible to open a blank draft.

```
cordova.plugins.email.open();
```

Send HTML encoded body

Its possible to send the email body either as text or HTML. In the case of HTML the `isHTML` properties needs to be set.

```
cordova.plugins.email.open({
    to:      'max@mustermann.de',
    subject: 'Greetings',
    body:    '<h1>Nice greetings from Leipzig</h1>',
    isHtml:  true
});
```

Get informed when the view has been dismissed

The `open` method supports additional callback to get informed when the view has been dismissed.

```
cordova.plugins.email.open(properties, function () {
    console.log('email view dismissed');
}, this);
```

Adding attachments

Attachments can be either base64 encoded datas, files from the the device storage or assets from within the *www* folder.

Attach Base64 encoded content

The code below shows how to attach an base64 encoded image which will be added as a image with the name *icon.png*.

```
cordova.plugins.email.open({
    subject:     'Cordova Icon',
    attachments: 'base64:icon.png//iVBORw0KGgoAAAANSUhEUgAAADwAAAA8CAYAAAA6/...'
});
```

Attach files from the device storage

The path to the files must be defined absolute from the root of the file system.

```
cordova.plugins.email.open({
    attachments: 'file:///storage/sdcard/icon.png', //=> Android
});
```

Attach native app resources

Each app has a resource folder, e.g. the *res* folder for Android apps or the *Resource* folder for iOS apps. The following example shows how to attach the app icon from within the app's resource folder.

```
cordova.plugins.email.open({
    attachments: 'res://icon.png' //=> res/drawable/icon (Android)
});
```

Attach assets from the www folder

The path to the files must be defined relative from the root of the mobile web app folder, which is located under the *www* folder.

```
cordova.plugins.email.open({
    attachments: [
        'file://img/logo.png', //=> assets/www/img/logo.png (Android)
        'file://css/index.css' //=> www/css/index.css (iOS)
    ]
});
```

Quirks

HTML and CSS on Android

Even Android is capable to render HTML formatted mails, most native Mail clients like the standard app or Gmail only support rich formatted text while writing mails. That means that **CSS cannot be used** (no *class* and *style* support).

The following table gives an overview which tags and attributes can be used:

• `<a href="...">`	• `<h4>`
• `<b>`	• `<h5>`
• `<big>`	• `<h6>`
• `<blockquote>`	• `<i>`
• `<br>`	• `<img src="...">`
• `<cite>`	• `<p>`
• `<dfn>`	• `<small>`
• `<div align="...">`	• `<strike>`
• `<em>`	• `<strong>`
• `<font size="..." color="..." face="...">`	• `<sub>`
• `<h1>`	• `<sup>`
• `<h2>`	• `<tt>`
• `<h3>`	• `<u>`

HTML and attachments on Windows Phone 8

Attachments and HTML formatted body are not supported through the native API.

Compile error on iOS

The error indicates, that the `MessageUI.framework` is not linked to your project. The framework is linked automatically when the plugin was installed, but may removed later.

```
Undefined symbols for architecture i386:
  "_OBJC_CLASS_$_MFMailComposeViewController", referenced from:
      objc-class-ref in APPEmailComposer.o
ld: symbol(s) not found for architecture i386
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

Contributing

1. Fork it
2. Create your feature branch (`git checkout -b my-new-feature`)
3. Commit your changes (`git commit -am 'Add some feature'`)
4. Push to the branch (`git push origin my-new-feature`)
5. Create new Pull Request

License

This software is released under the Apache 2.0 License.