buzztouch™

BuzzTouch URL Scheme

Niraj Shah

September 2, 2013
Version 1.0

BuzzTouch URL Scheme

Additons to the PLIST file

Add URL Scheme properties into BT_Info.plist file

The scheme name (or protocol) of a URL is the first part of a URL - e.g. s**chemename://**.

iOS supports these **built-in** URL schemes:

- **http, https, ftp** (launches Safari)
- **mailto** Email (launches the Mail app)
- **tel** Telephone Numbers (launches the phone app)
- **sms** Text Messages (launches the SMS app)

A few automatic redirects are pre-programmed into iOS:

- Web links that point to http://maps.google.com are redirected to the Maps app.
- Web links that point to http://www.youtube.com are redirected to the YouTube app.
- iTunes store links (https://itunes.apple.com/us/app..) are sent to the iTunes (or App store) app

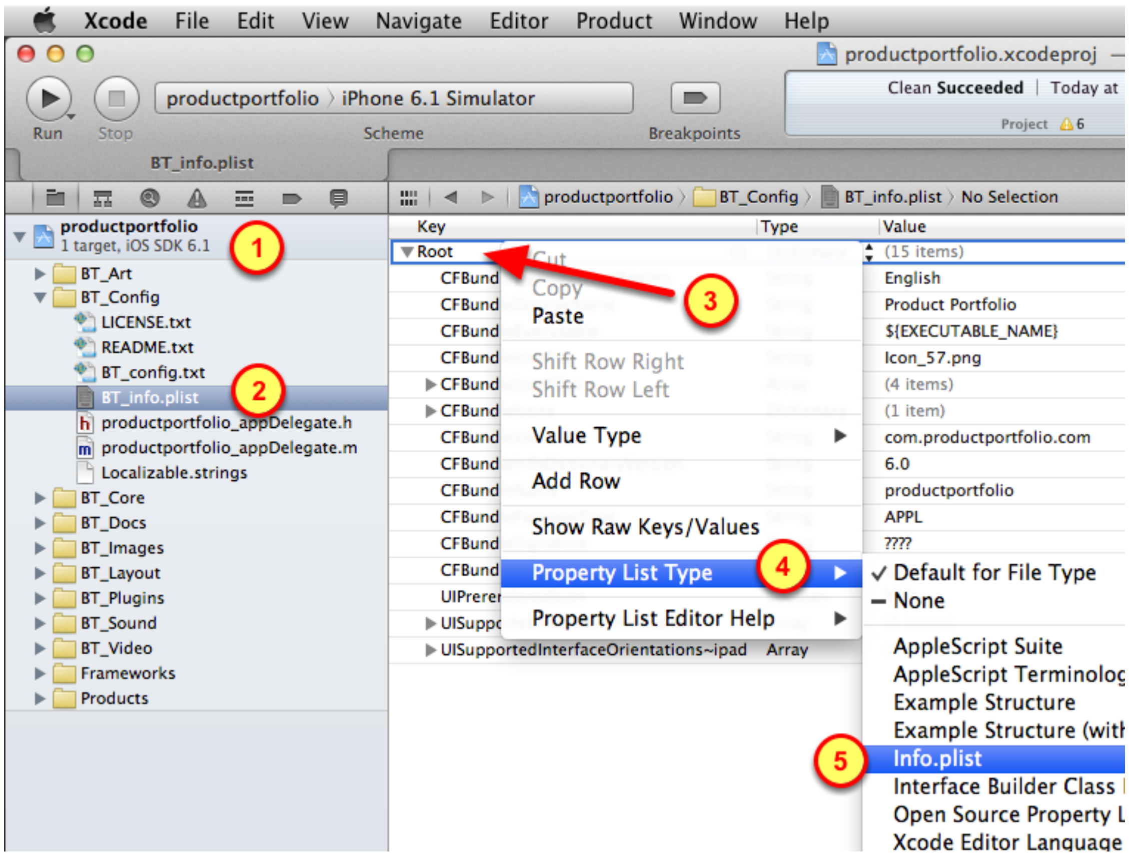
iOS apps can also specify their own custom URL scheme (for example, myapp://doStuff). When might you want to use a custom URL scheme for your app?

- To transfer data from lite to paid versions of your app
- To allow other apps (or even web pages) to call your app (and send data to it)
- To handle callbacks for custom authentication (such as OAuth) and third party API's

Defining your app's custom URL scheme is all done in the Info.plist file.

Most of this content in this introductory section comes from the terrific iDev101.com web site on Custom URL Schemes:
http://www.idev101.com/code/Objective-C/custom_url_schemes.html

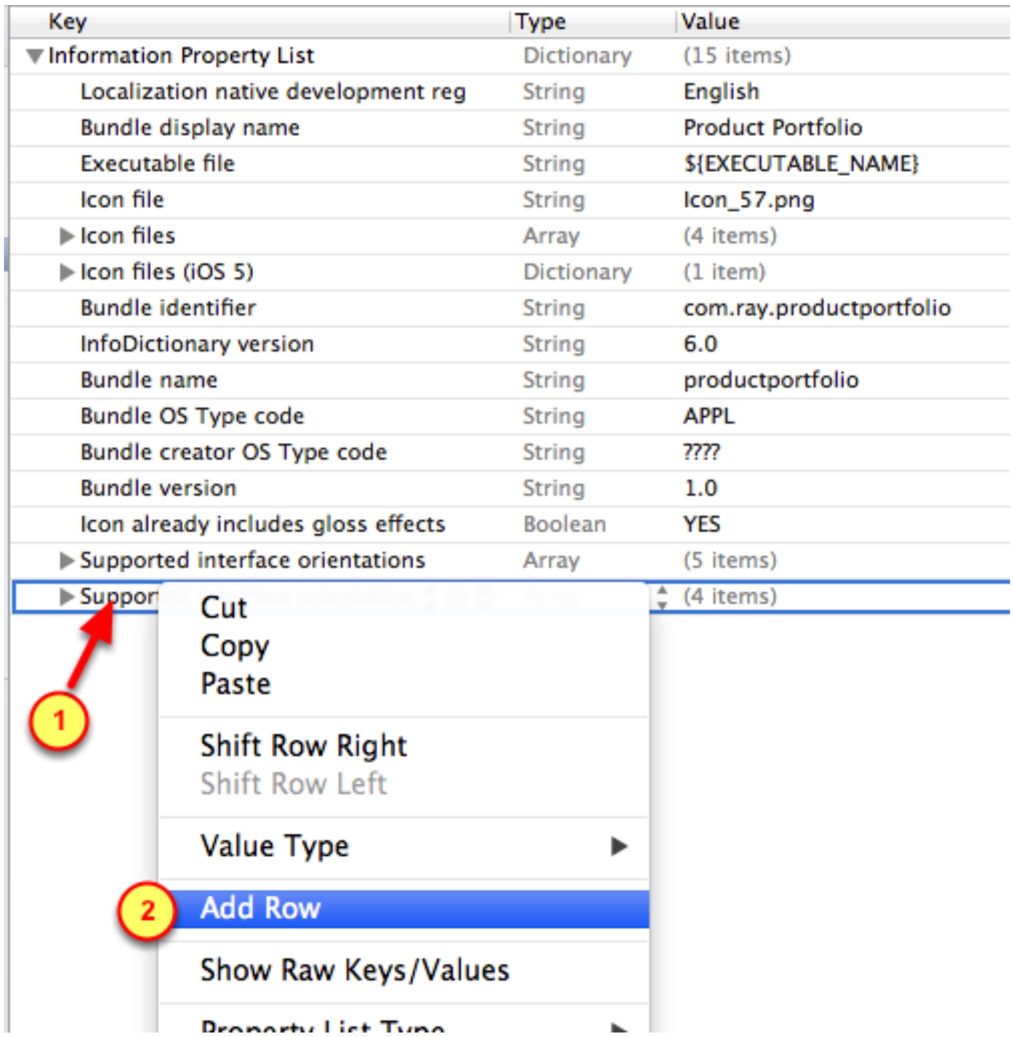
1. Change to an Info.plist document type



To change the plist file to be recognized as an Info.plist property list type:

1. Open up the **project**
2. Open the BT_Config group and select the **BT_info.plist** file
3. Right-click on the **Root** row
4. Select the **Property List Type** menu item
5. Choose the **Info.plist** type

2. Add a new row



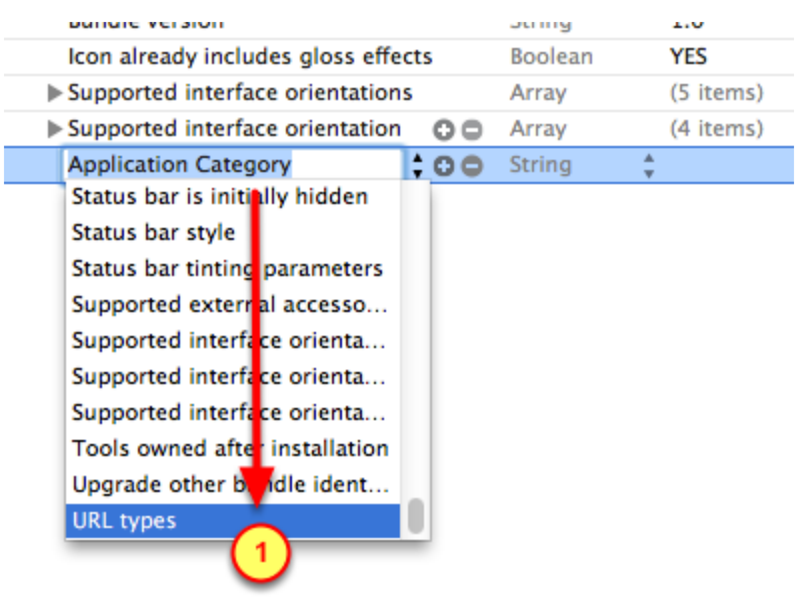
Key	Type	Value
▼ Information Property List	Dictionary	(15 items)
Localization native development reg	String	English
Bundle display name	String	Product Portfolio
Executable file	String	${EXECUTABLE_NAME}
Icon file	String	Icon_57.png
▶ Icon files	Array	(4 items)
▶ Icon files (iOS 5)	Dictionary	(1 item)
Bundle identifier	String	com.ray.productportfolio
InfoDictionary version	String	6.0
Bundle name	String	productportfolio
Bundle OS Type code	String	APPL
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Icon already includes gloss effects	Boolean	YES
▶ Supported interface orientations	Array	(5 items)
▶ Suppor		(4 items)

To add a new row into the BT_info.plist file

1. On the **last row** of the file, right-click

2. Select the **Add Row** menu item

3. Specify URL types as the property type



Icon already includes gloss effects	Boolean	YES
▶ Supported interface orientations	Array	(5 items)
▶ Supported interface orientation	Array	(4 items)
Application Category	String	

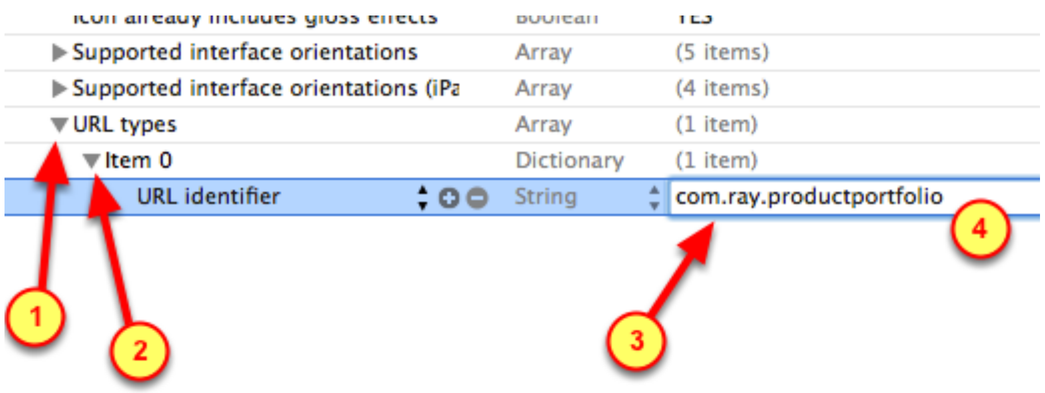
Status bar is initially hidden
Status bar style
Status bar tinting parameters
Supported external accesso…
Supported interface orienta…
Supported interface orienta…
Supported interface orienta…
Tools owned after installation
Upgrade other bundle ident…
URL types
1

Specify the new property is a URL

1. Scroll-down to the **bottom** of the list

2. Select **URL types** as the property type for the newly added row

4. Specify a URL identifier



▶ Supported interface orientations	Array	(5 items)
▶ Supported interface orientations (iPa	Array	(4 items)
▼ URL types	Array	(1 item)
▼ Item 0	Dictionary	(1 item)
URL identifier	String	com.ray.productportfolio

Specify the URL Identifier:

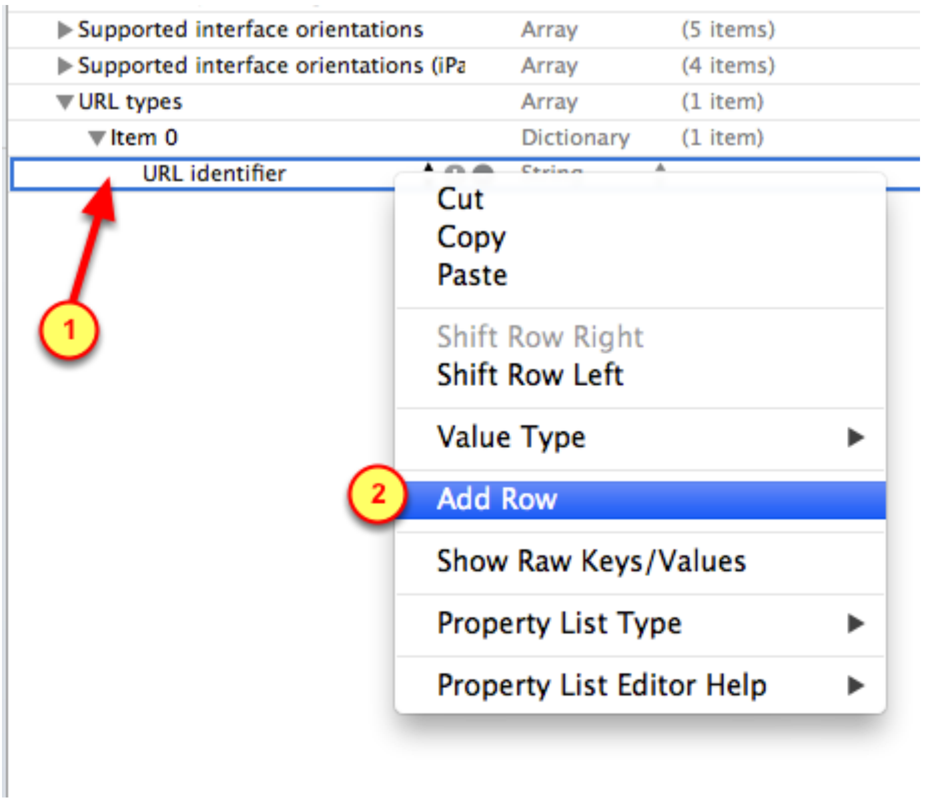
1. Expand the **URL types** row

2. Expand the **Item 0** child row

3. Select on the **URL Identifier** row, double-click in the value box for that row

4. Specify the same **name** as what is entered for the Bundle Identifier (see previous rows in same file)

5. Add a new row for URL Schemes

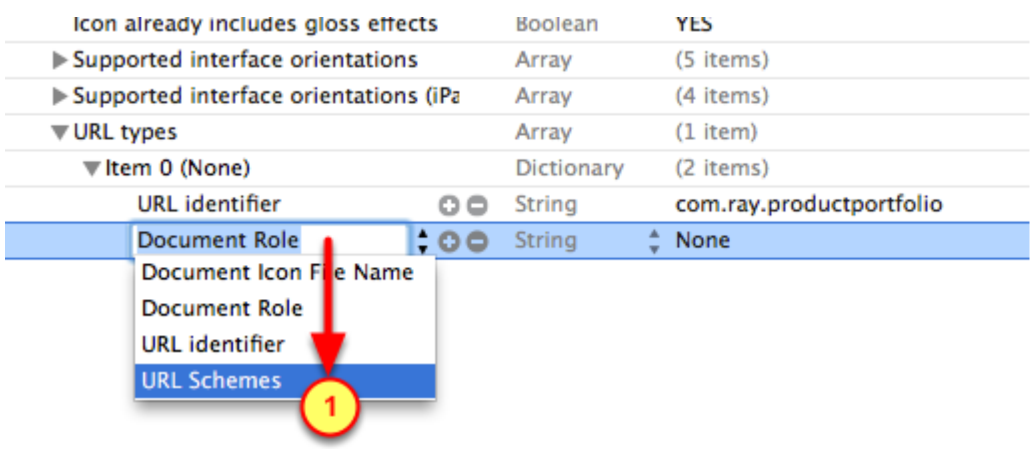

Supported interface orientations | Array | (5 items)
Supported interface orientations (iPa | Array | (4 items)
URL types | Array | (1 item)
Item 0 | Dictionary | (1 item)
URL identifier | String
Cut
Copy
Paste
Shift Row Right
Shift Row Left
Value Type
Add Row
Show Raw Keys/Values
Property List Type
Property List Editor Help
1
2

To add a new row into the BT_info.plist file

1. On the **last row** of the file, right-click

2. Select the **Add Row** menu item

6. Specify URL Schemes as the property type



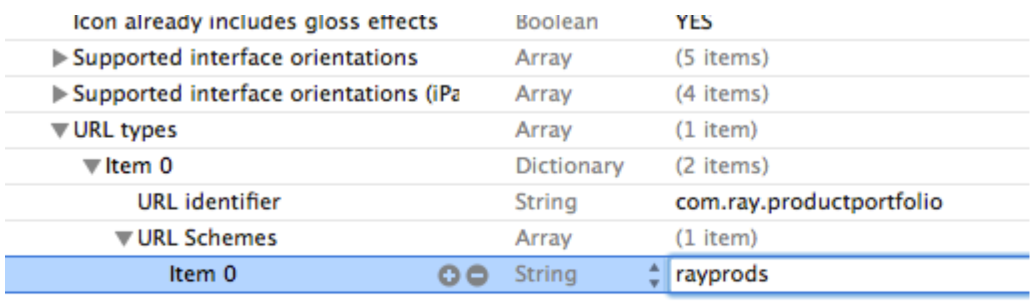
Icon already includes gloss effects	Boolean	YES
▶ Supported interface orientations	Array	(5 items)
▶ Supported interface orientations (iPa	Array	(4 items)
▼ URL types	Array	(1 item)
▼ Item 0 (None)	Dictionary	(2 items)
URL identifier	String	com.ray.productportfolio
Document Role	String	None

Specify the new property is a URL Schemes

1. Scroll-down to the **bottom** of the list

2. Select **URL Schemes** as the property type for the newly added row

7. Specify the URL Scheme



Icon already includes gloss effects	Boolean	YES
▶ Supported interface orientations	Array	(5 items)
▶ Supported interface orientations (iPa	Array	(4 items)
▼ URL types	Array	(1 item)
▼ Item 0	Dictionary	(2 items)
URL identifier	String	com.ray.productportfolio
▼ URL Schemes	Array	(1 item)
Item 0	String	rayprods

Specify the URL Scheme, it is the part that comes before the :// characters in a URL.

An example of a URL Scheme is "myCoolApp" as in **myCoolApp**://myCoolScreenNickname

1. Expand the **URL Schemes** row

2. Click on the **Item 0** row

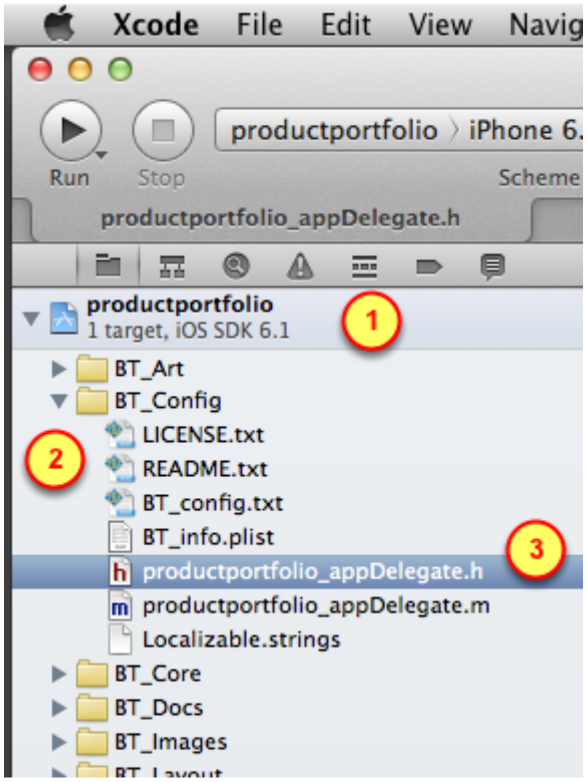
3. Double-click in the **value box** for that row

4. Specify a **short** and **unique** name for the URL Scheme

Code to put in the App Delegate files

Declarations for App Delegate's Header file (.h)

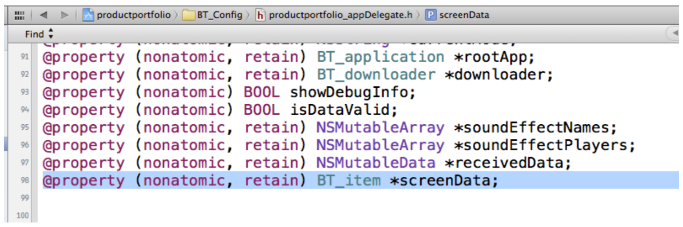
1. Locate and open App Delegate's Header file



Find and edit the App Delegate's Header file (.h)

1. Open the **Project**
2. Open the **BT_Config** group
3. Click on the **App Delegate's header file** (.h)

Insert Property into Header file

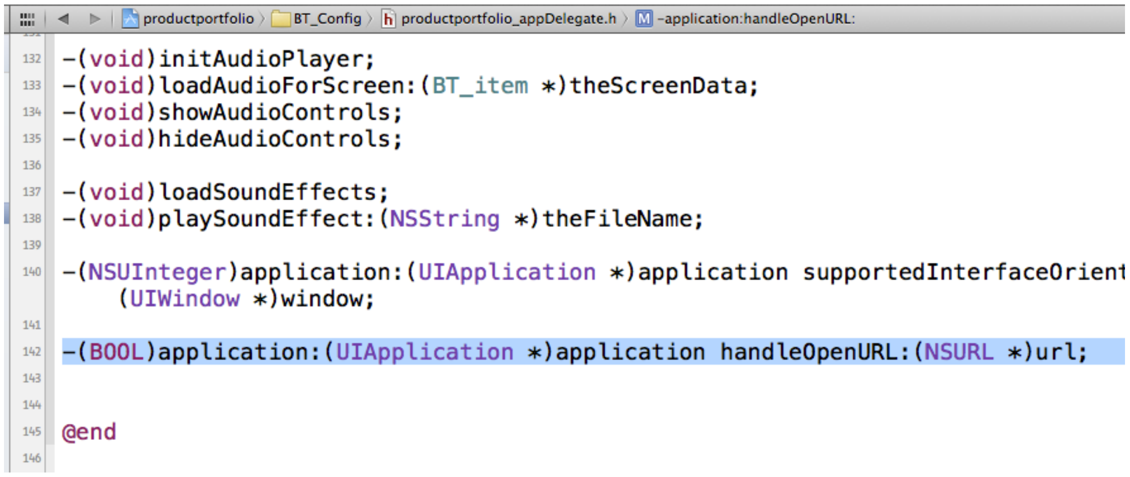


```
@property (nonatomic, retain) BT_application *rootApp;
@property (nonatomic, retain) BT_downloader *downloader;
@property (nonatomic) BOOL showDebugInfo;
@property (nonatomic) BOOL isDataValid;
@property (nonatomic, retain) NSMutableArray *soundEffectNames;
@property (nonatomic, retain) NSMutableArray *soundEffectPlayers;
@property (nonatomic, retain) NSMutableData *receivedData;
@property (nonatomic, retain) BT_item *screenData;
```

At the end of the Property block in the App Delegate's header file (.h), insert this line:

```
@property (nonatomic, retain) BT_item *screenData;
```

Insert Declaration into Header file



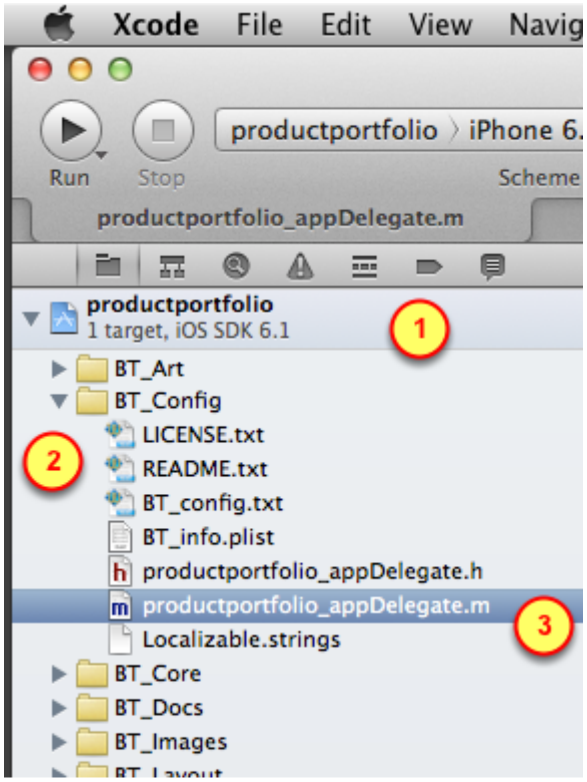
```
-(void)initAudioPlayer;
-(void)loadAudioForScreen:(BT_item *)theScreenData;
-(void)showAudioControls;
-(void)hideAudioControls;

-(void)loadSoundEffects;
-(void)playSoundEffect:(NSString *)theFileName;

-(NSUInteger)application:(UIApplication *)application supportedInterfaceOrient
    (UIWindow *)window;

-(BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url;


@end
```

At the end of the App Delegate's header file (.h), insert this line:

```
-(BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url;
```

Code for the App Delegate's Implementation file (.m)

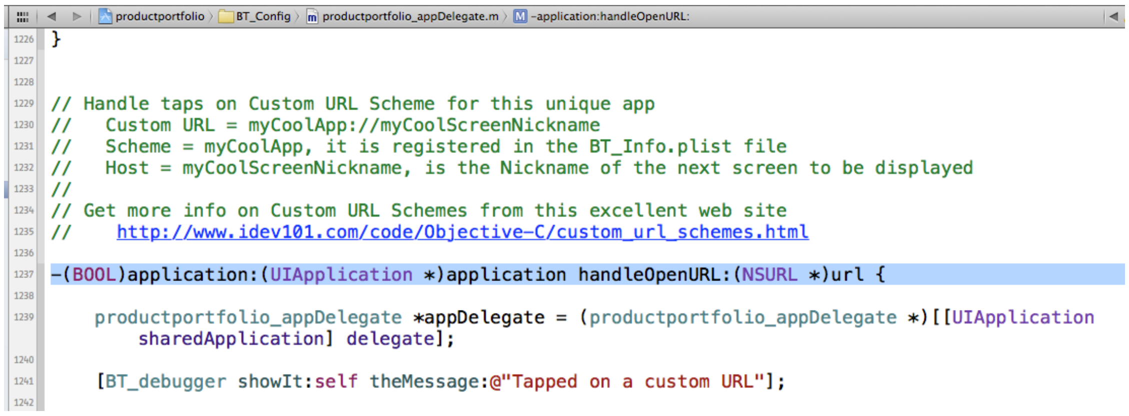
Open the App Delegate's Implementation file (.m)



Find and edit the App Delegate's Implementation file (.m)

1. Open the **Project**
2. Open the **BT_Config** group
3. Click on the **App Delegate's implementation file** (.m)

Insert Handler into Implementation file (.m)

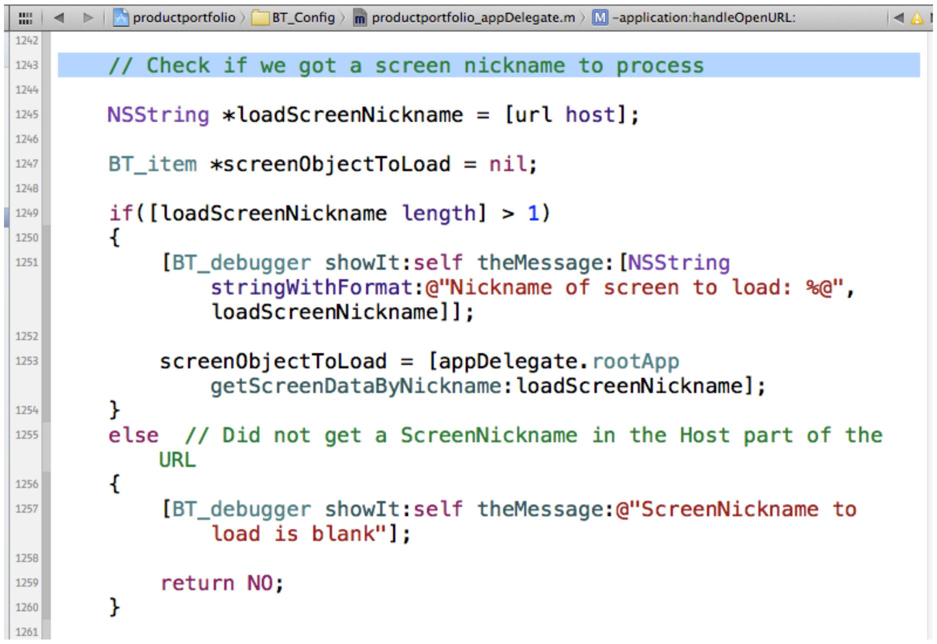


```
}


// Handle taps on Custom URL Scheme for this unique app
//   Custom URL = myCoolApp://myCoolScreenNickname
//   Scheme = myCoolApp, it is registered in the BT_Info.plist file
//   Host = myCoolScreenNickname, is the Nickname of the next screen to be displayed
//
// Get more info on Custom URL Schemes from this excellent web site
//    http://www.idev101.com/code/Objective-C/custom_url_schemes.html

-(BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url {

    productportfolio_appDelegate *appDelegate = (productportfolio_appDelegate *)[[UIApplication
        sharedApplication] delegate];

    [BT_debugger showIt:self theMessage:@"Tapped on a custom URL"];

```

Just before the dealloc method, insert the Handler for the Custom URL Scheme:

```
-(BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url {

    productportfolio_appDelegate *appDelegate =
(productportfolio_appDelegate *)[[UIApplication sharedApplication]
delegate];

    [BT_debugger showIt:self theMessage:@"Tapped on a custom URL"];
```

Check if a Screen Nickname was provided

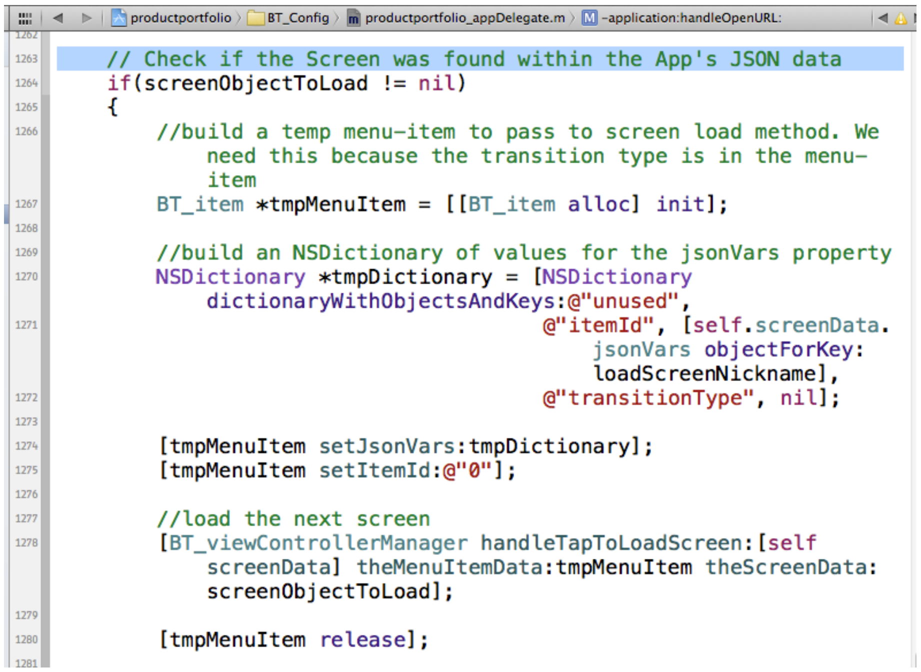


```

    // Check if we got a screen nickname to process

    NSString *loadScreenNickname = [url host];

    BT_item *screenObjectToLoad = nil;

    if([loadScreenNickname length] > 1)
    {
        [BT_debugger showIt:self theMessage:[NSString
            stringWithFormat:@"Nickname of screen to load: %@",
            loadScreenNickname]];

        screenObjectToLoad = [appDelegate.rootApp
            getScreenDataByNickname:loadScreenNickname];
    }
    else   // Did not get a ScreenNickname in the Host part of the
        URL
    {
        [BT_debugger showIt:self theMessage:@"ScreenNickname to
            load is blank"];

        return NO;
    }

```

Ensure a screen nickname is in the Host field of the URL:

```
    NSString *loadScreenNickname = [url host];
    BT_item *screenObjectToLoad = nil;

    // Check if we got a screen nickname to process
    if([loadScreenNickname length] > 1)
    {
        [BT_debugger showIt:self theMessage:[NSString
stringWithFormat:@"Nickname of screen to load: %@",loadScreenNickname]];

        screenObjectToLoad = [appDelegate.rootApp
getScreenDataByNickname:loadScreenNickname];
    }
    else  // Did not get a ScreenNickname in the Host part of the URL
    {
        [BT_debugger showIt:self theMessage:@"ScreenNickname to load is
blank"];

        return NO;
    }
```

Check if the Screen exists and Load that Screen



```
    // Check if the Screen was found within the App's JSON data
    if(screenObjectToLoad != nil)
    {
        //build a temp menu-item to pass to screen load method. We
            need this because the transition type is in the menu-
            item
        BT_item *tmpMenuItem = [[BT_item alloc] init];

        //build an NSDictionary of values for the jsonVars property
        NSDictionary *tmpDictionary = [NSDictionary
            dictionaryWithObjectsAndKeys:@"unused",
                                    @"itemId", [self.screenData.
                                        jsonVars objectForKey:
                                        loadScreenNickname],
                                    @"transitionType", nil];

        [tmpMenuItem setJsonVars:tmpDictionary];
        [tmpMenuItem setItemId:@"0"];

        //load the next screen
        [BT_viewControllerManager handleTapToLoadScreen:[self
            screenData] theMenuItemData:tmpMenuItem theScreenData:
            screenObjectToLoad];

        [tmpMenuItem release];
```

Four part sequence:

1. Check if the Screen Nickname led us to an actual screen that has data in the BT_config.txt file

(the JSON data)

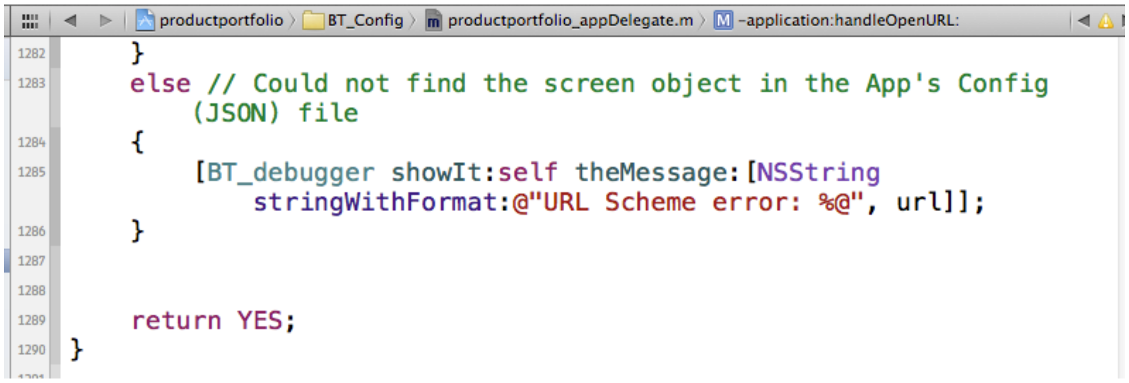
2. Create a fake set of temporary data for that Screen

3. Load that screen using that fake temp data

4. Give back the memory space of that temporary data back for other usages

```
    // Check if the Screen was found within the App's JSON data
    if(screenObjectToLoad != nil)
    {
        // Build a temp menu-item to pass to screen load method.
        // We need this because the transition type is in the menu-item
        BT_item *tmpMenuItem = [[BT_item alloc] init];

        // Build an NSDictionary of values for the jsonVars property
        NSDictionary *tmpDictionary = [NSDictionary
dictionaryWithObjectsAndKeys:@"unused",
                                        @"itemId", [self.screenData.jsonVars
objectForKey:loadScreenNickname],
                                        @"transitionType", nil];

        [tmpMenuItem setJsonVars:tmpDictionary];
        [tmpMenuItem setItemId:@"0"];

        // Load the next screen
        [BT_viewControllerManager handleTapToLoadScreen:[self screenData]
theMenuItemData:tmpMenuItem theScreenData:screenObjectToLoad];

        [tmpMenuItem release];
```

Error if cannot find Screen's data



```
    }
    else // Could not find the screen object in the App's Config
        (JSON) file
    {
        [BT_debugger showIt:self theMessage:[NSString
            stringWithFormat:@"URL Scheme error: %@", url]];
    }


    return YES;
}
```

If the Screen does not have data in the BT_config.txt file (the JSON data), then report an error:

```
    }
    else // Could not find the screen object in the App's Config (JSON) file
    {
        [BT_debugger showIt:self theMessage:[NSString
stringWithFormat:@"URL Scheme error: %@", url]];
    }


    return YES;
}
```

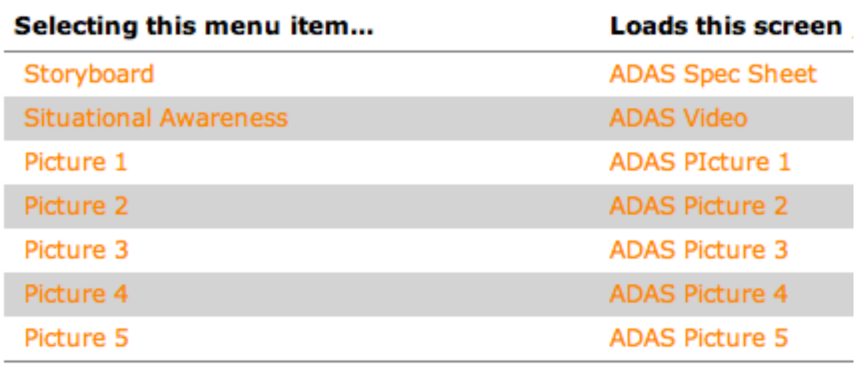
Entire code block in the Implementation file

```
// Handle taps on Custom URL Scheme for this unique app
//   Custom URL = myCoolApp://myCoolScreenNickname
//   Scheme = myCoolApp, it is registered in the BT_Info.plist file
//   Host = myCoolScreenNickname, is the Nickname of the next screen to be
displayed
//
// Get more info on Custom URL Schemes from this excellent web site
//    http://www.idev101.com/code/Objective-C/custom_url_schemes.html

-(BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url {

    productportfolio_appDelegate *appDelegate =
(productportfolio_appDelegate *)[[UIApplication sharedApplication]
```

```
delegate];

    [BT_debugger showIt:self theMessage:@"Tapped on a custom URL"];

    // Check if we got a screen nickname to process

    NSString *loadScreenNickname = [url host];

    BT_item *screenObjectToLoad = nil;

    if([loadScreenNickname length] > 1)
    {
        [BT_debugger showIt:self theMessage:[NSString
stringWithFormat:@"Nickname of screen to load: %@",loadScreenNickname]];

        screenObjectToLoad = [appDelegate.rootApp
getScreenDataByNickname:loadScreenNickname];
    }
    else  // Did not get a ScreenNickname in the Host part of the URL
    {
        [BT_debugger showIt:self theMessage:@"ScreenNickname to load is
blank"];

        return NO;
    }


    // Check if the Screen was found within the App's JSON data
    if(screenObjectToLoad != nil)
    {
        // Build a temp menu-item to pass to screen load method.
        // We need this because the transition type is in the menu-item
        BT_item *tmpMenuItem = [[BT_item alloc] init];

        // Build an NSDictionary of values for the jsonVars property
        NSDictionary *tmpDictionary = [NSDictionary
dictionaryWithObjectsAndKeys:@"unused",
                                       @"itemId", [self.screenData.jsonVars
objectForKey:loadScreenNickname],
                                       @"transitionType", nil];

        [tmpMenuItem setJsonVars:tmpDictionary];
        [tmpMenuItem setItemId:@"0"];
        // Load the next screen
        [BT_viewControllerManager handleTapToLoadScreen:[self screenData]
theMenuItemData:tmpMenuItem theScreenData:screenObjectToLoad];

        [tmpMenuItem release];

    }
    else // Could not find the screen object in the App's Config (JSON) file
    {
        [BT_debugger showIt:self theMessage:[NSString
stringWithFormat:@"URL Scheme error: %@", url]];
    }


    return YES;
}
```

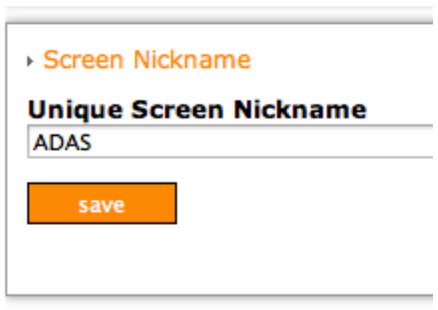
Testing the App

Make an ADAS screen



Selecting this menu item...	Loads this screen
Storyboard	ADAS Spec Sheet
Situational Awareness	ADAS Video
Picture 1	ADAS PIcture 1
Picture 2	ADAS Picture 2
Picture 3	ADAS Picture 3
Picture 4	ADAS Picture 4
Picture 5	ADAS Picture 5

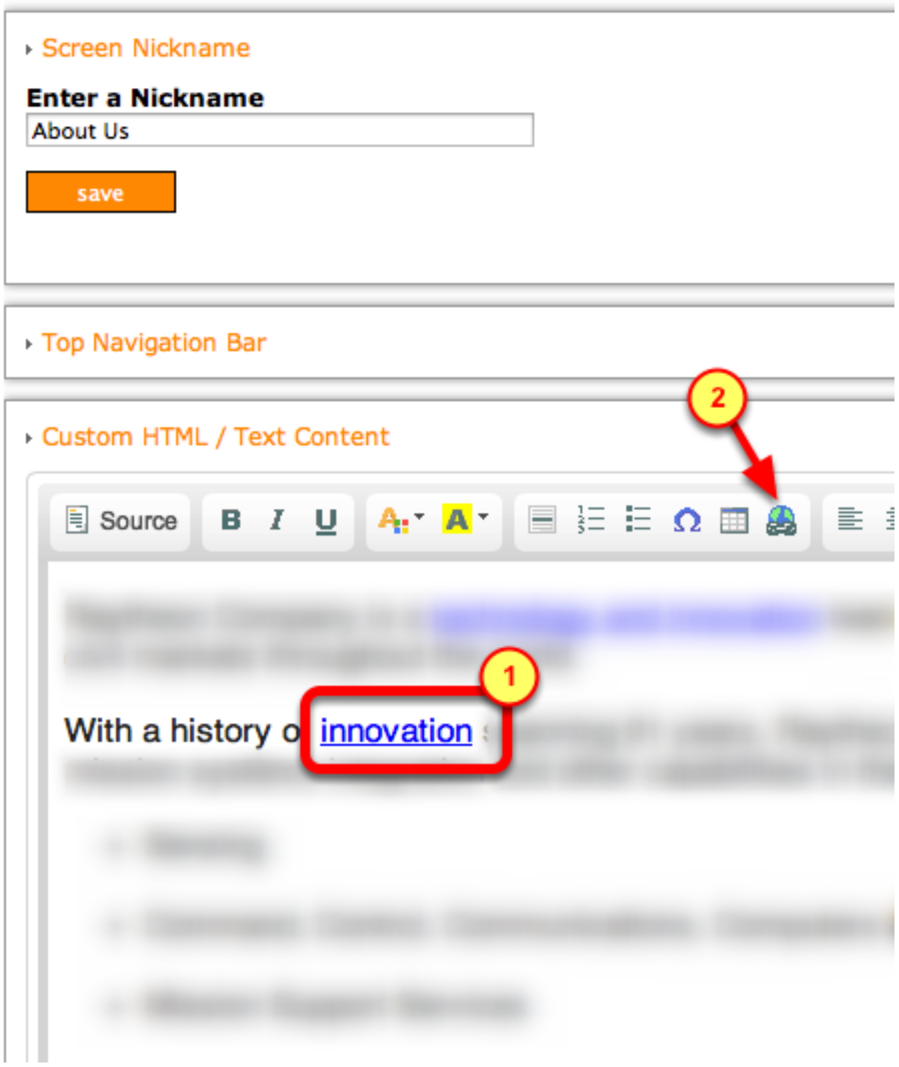
Make an ADAS screen nickname



Screen Nickname

Unique Screen Nickname
ADAS

Make an About Us screen

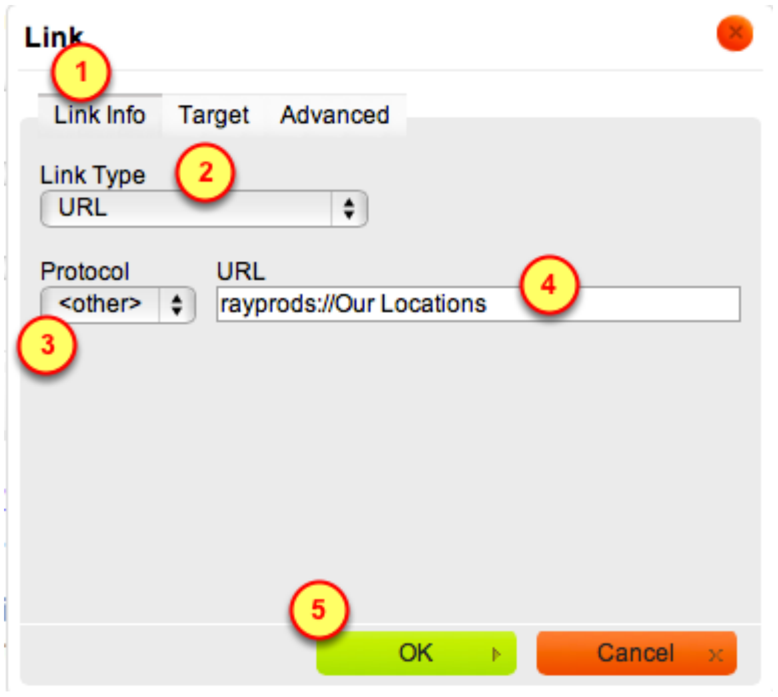


Hyperlink words to point to another screen within this app.

1. Select and highlight the words

2. Click on the **Link Tool** to specify the custom URL

Add a custom URL link



Link Type: URL

Protocol: <other>

URL: rayprods://Our Locations

Specify this is a custom URL

1. Be on the **Link Info** tab

2. The **Link Type** should be of type **URL**

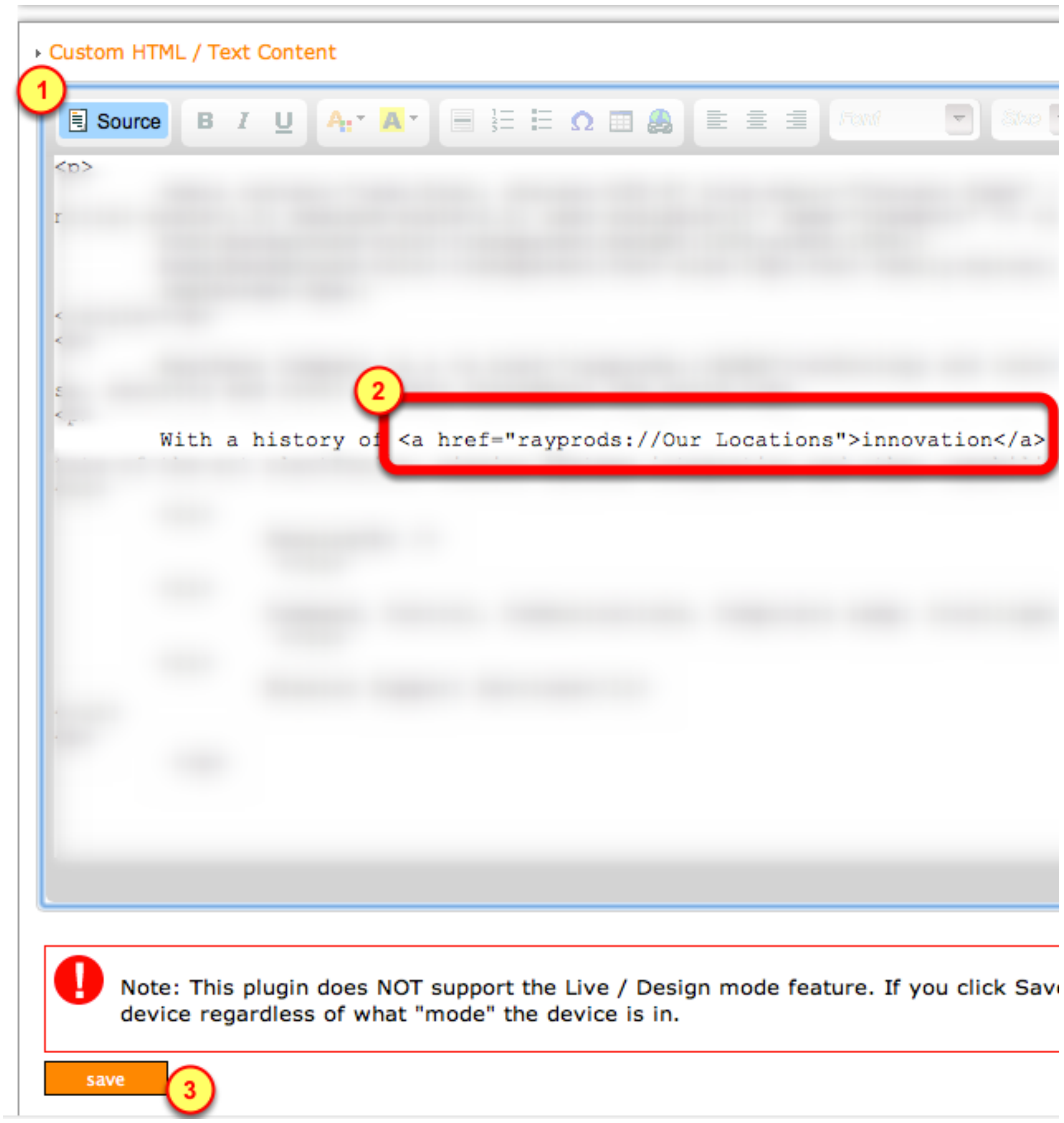
3. The **Protocol** should be **<other>**

4. The **URL** value should be the **Custom URL**

* Spaces are allowed in the custom URL, make an exact match for the Screen Nickname)

5. Click on the **OK** button to save the entry for the custom URL

Inspect the custom link



```
<p>
With a history of <a href="rayprods://Our Locations">innovation</a>
```

Note: This plugin does NOT support the Live / Design mode feature. If you click Sav
device regardless of what "mode" the device is in.

View the source code to validate the HTML is correct

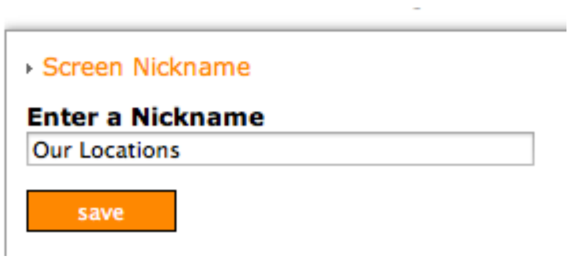
1. Click on the **Source** button to see the actual HTML

2. Verify the Custom URL looks like this:

```
<a href="rayprods://Our Locations">innovation</a>
```

3. Click on the **save** button

Make screen for Our Locations

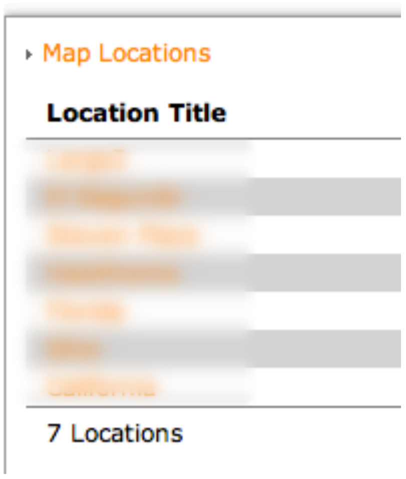


Screen Nickname

Enter a Nickname

Our Locations

Add map locations for Our Locations screen



Map Locations

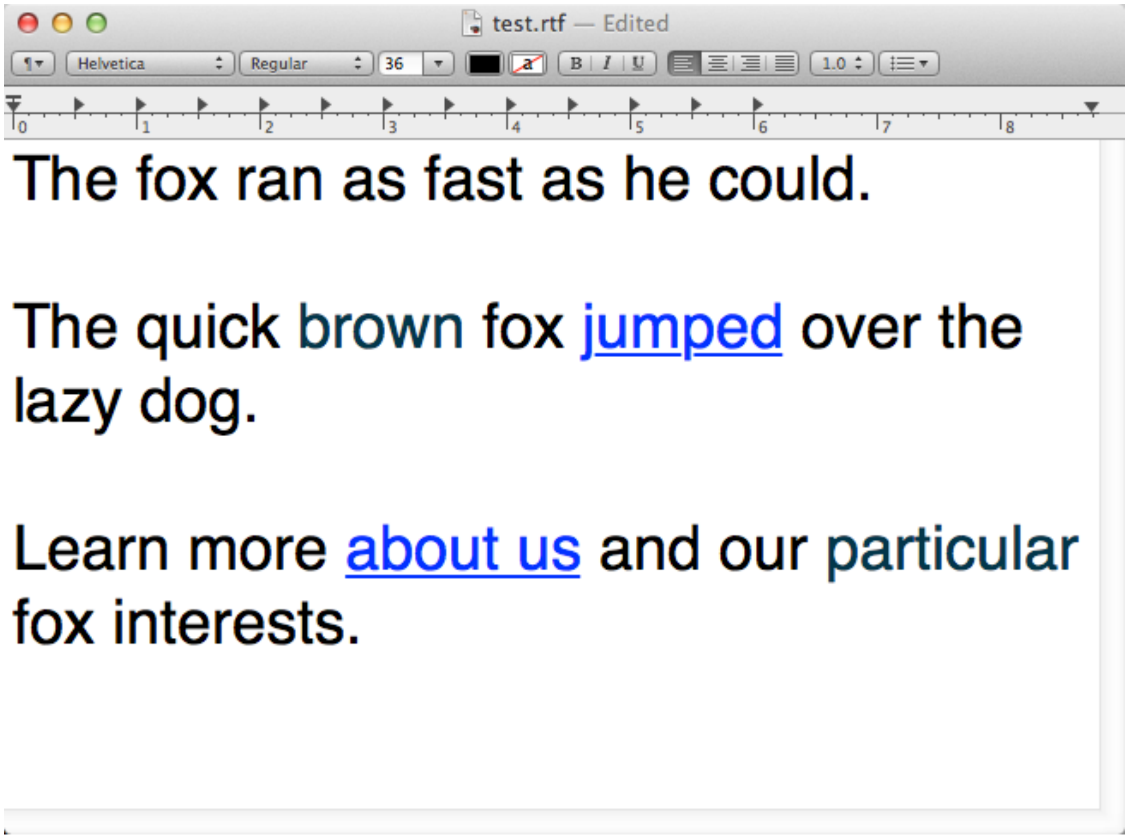
Location Title

[redacted]

7 Locations

Create a PDF test file

Create a text file



The fox ran as fast as he could.

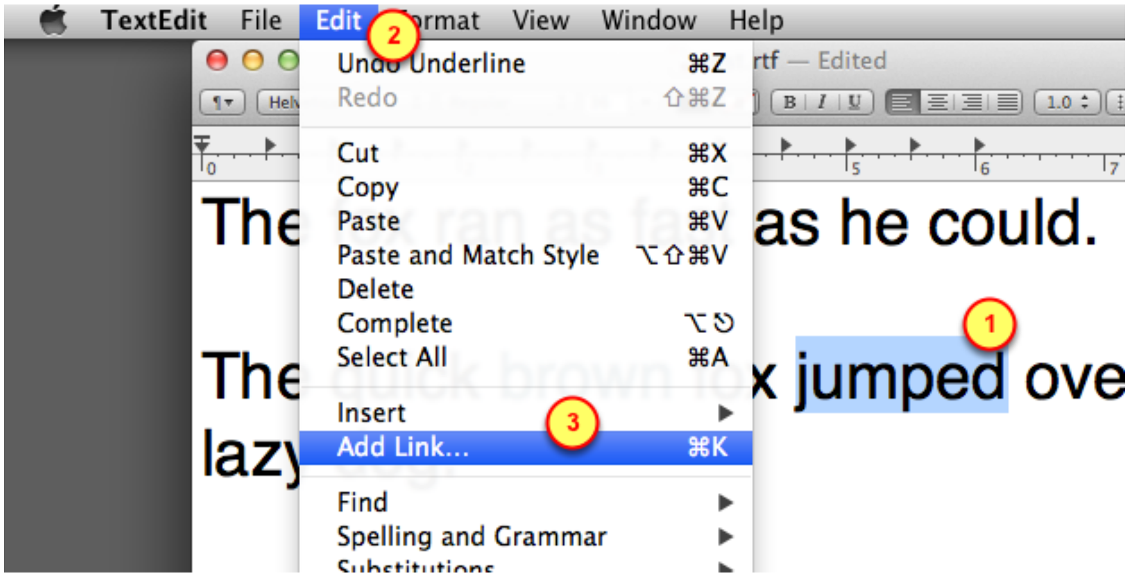
The quick brown fox jumped over the lazy dog.

Learn more about us and our particular fox interests.

Using the TextEdit application on the Mac, create a test document with sample text:

```
The fox ran as fast as he could.

The quick brown fox jumped over the lazy dog.

Learn more about us and our particular fox interests.
```

Create custom links in the text document



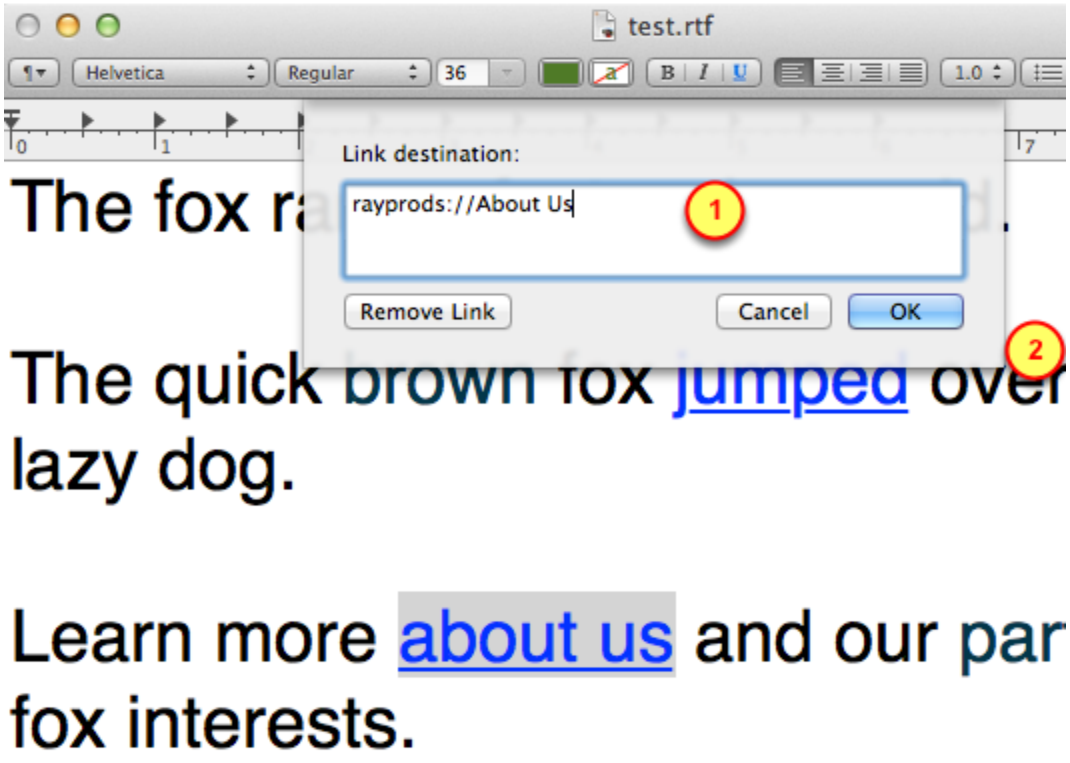
Lets hyperlink one of the words for the Custom URL link:

1. Select and **highlight** a word to be hyperlinked

2. Click on the **Edit** menu

3. Select the **Add Link** menu item

Specify the custom URL



The fox r…

The quick brown fox jumped over lazy dog.

Learn more about us and our par… fox interests.

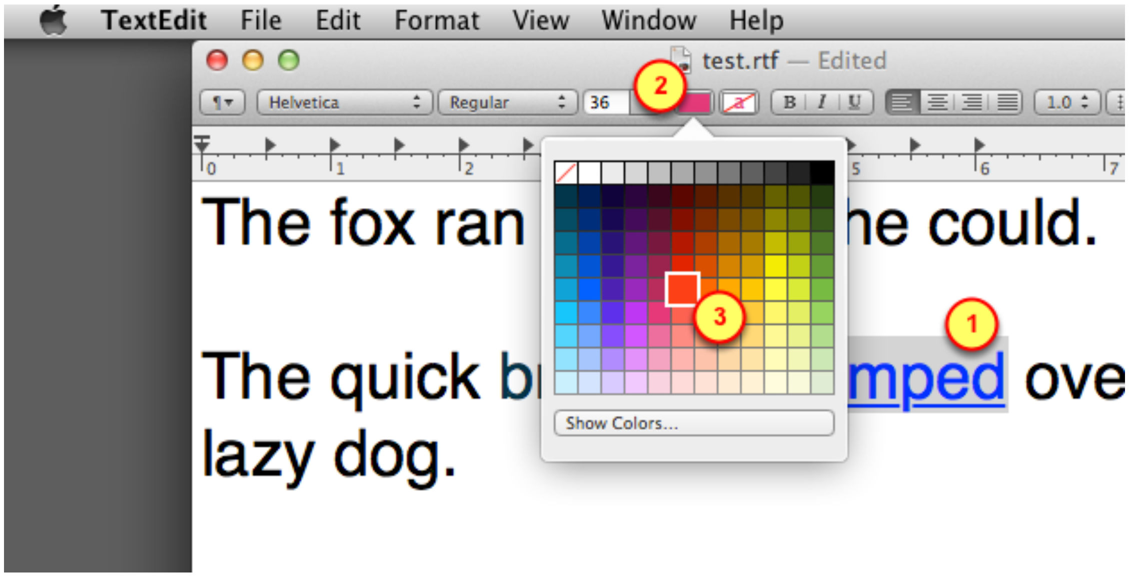
Insert the custom URL in the dialog box:

1. Specify the link for the **custom URL**.

   Remember that it must have two components:
   - **Scheme**: rayprods
   - **Screen Nickname**: About Us (it is okay to have spaces in the nickname)

2. Click on **OK** to complete this step

Color the link to look like a link



The fox ran ... he could.

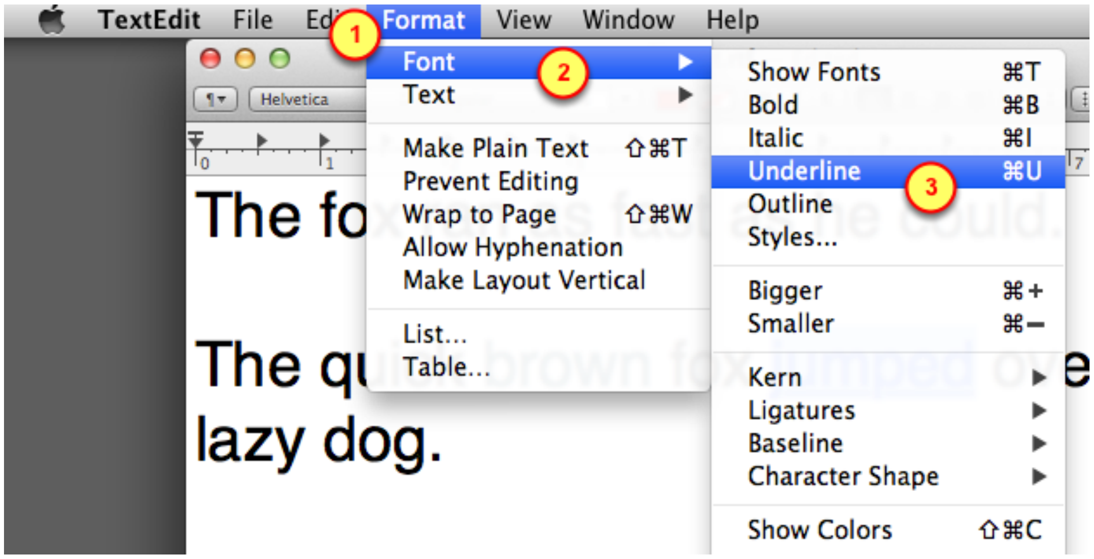
The quick b... mped ove... lazy dog.

Even though the link looks like a link within the text editor, it won't look like it as a PDF. Lets color that link by changing it's color.

1. Select and **highlight** the word

2. Using the **color box**, select a **color** for the word

Underline the link to look like a link

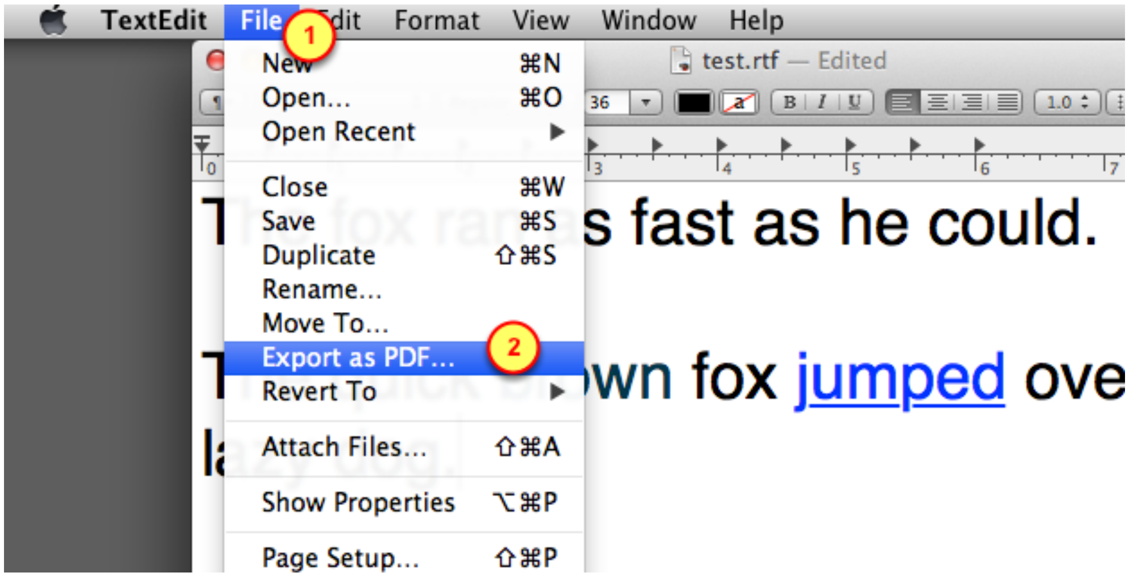


Lets underline the link explicitly, even though it already looks like an underlined link:

0. Select and **highlight** the word

1. Click on the **Format** menu
2. Click on the **Font** menu
3. Select the **Underline** menu item

Print a PDF version of the text file



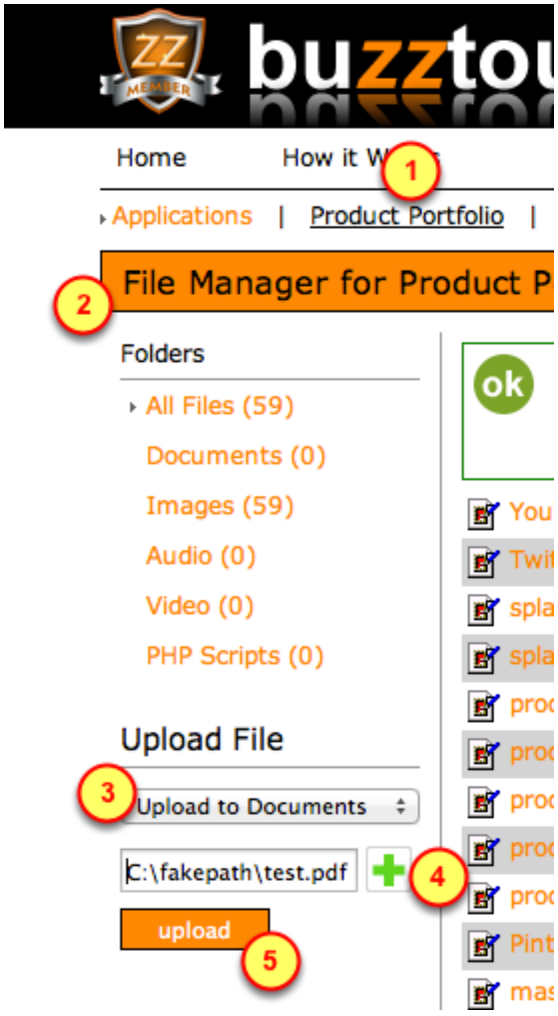
The fox ran as fast as he could.

The quick brown fox jumped over the lazy dog.

Generate a PDF version of the test file

1. Click on the **File** menu
2. Select the **Export as PDF** menu item

Upload the PDF file to BuzzTouch server



Upload the test PDF file into the BuzzTouch server for your App

1. Select the **Application** being tested

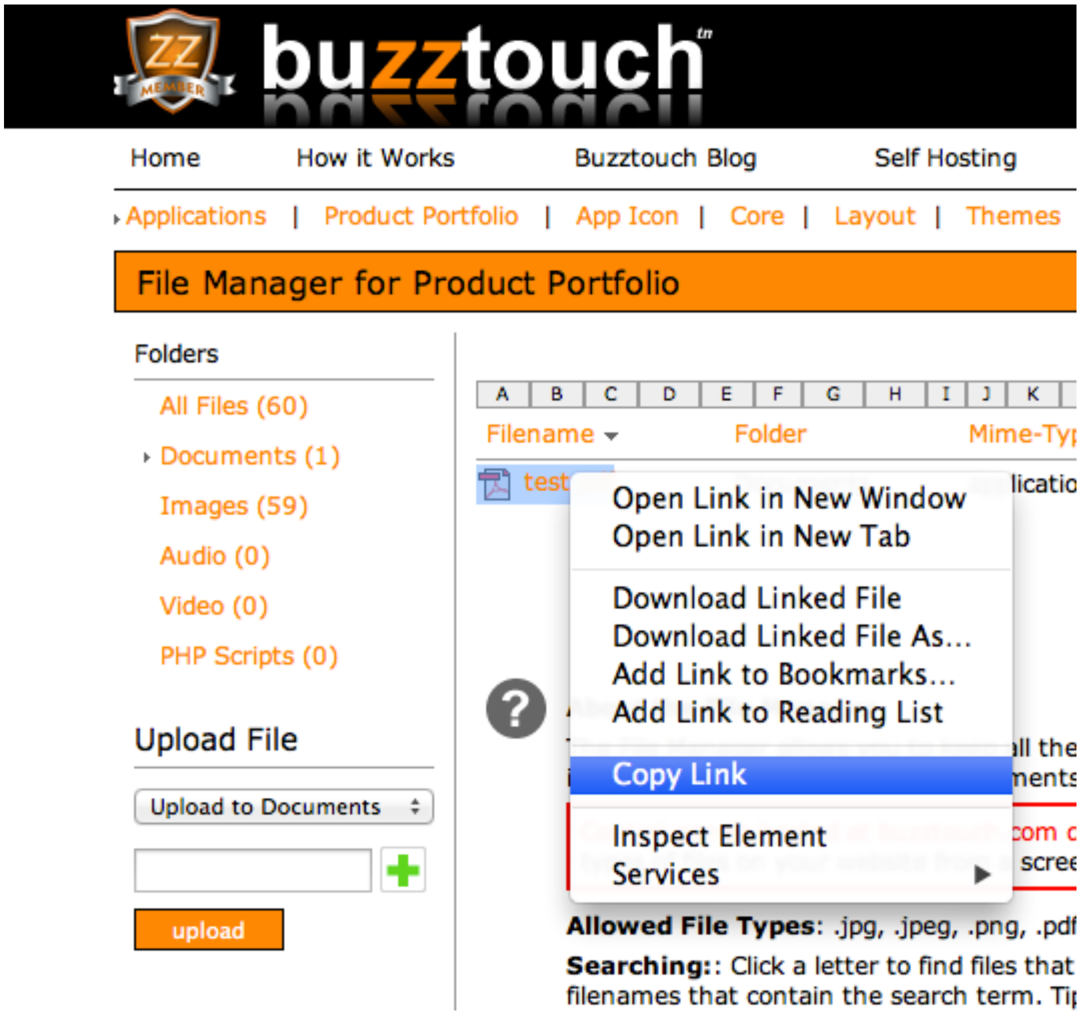
2. Select the **Files** section of the Application

3. Select **Documents** as the file type

4. **Browse** to the PDF file

5. Click on the **upload** button

Copy link for PDF file

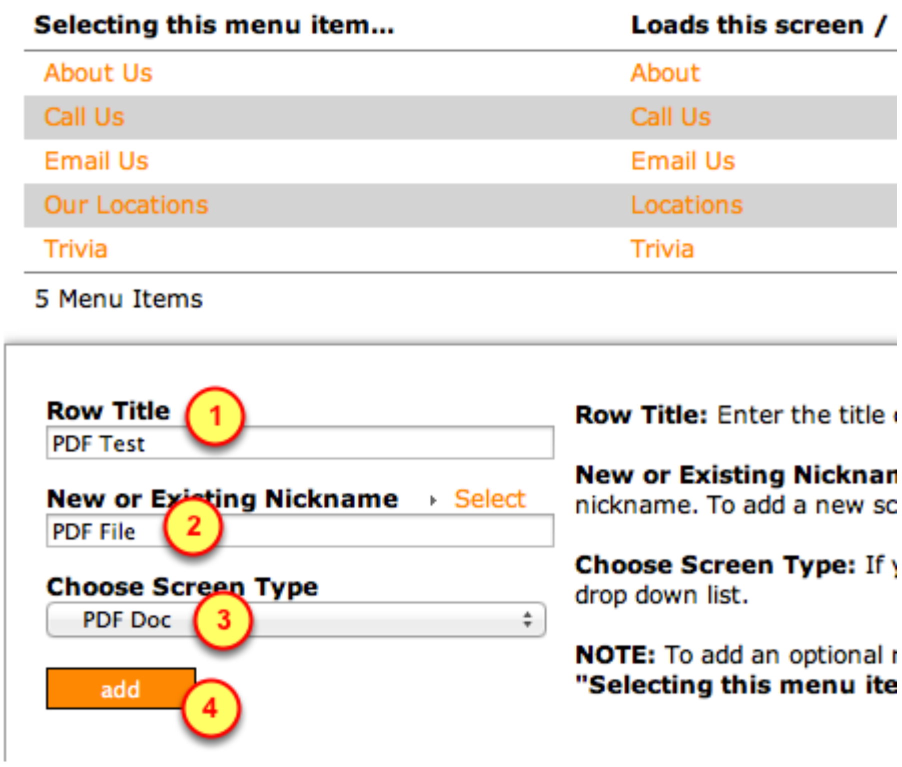


Allowed File Types: .jpg, .jpeg, .png, .pdf

Searching:: Click a letter to find files that
filenames that contain the search term. Ti

Get the link to the PDF file, it will be used to as a sample screen.

Add a BuzzTouch menu item pointing to PDF file



Selecting this menu item...	Loads this screen /
About Us	About
Call Us	Call Us
Email Us	Email Us
Our Locations	Locations
Trivia	Trivia

5 Menu Items

Row Title: PDF Test

New or Existing Nickname ‣ Select: PDF File

Choose Screen Type: PDF Doc

add

Row Title: Enter the title

New or Existing Nicknar nickname. To add a new sc

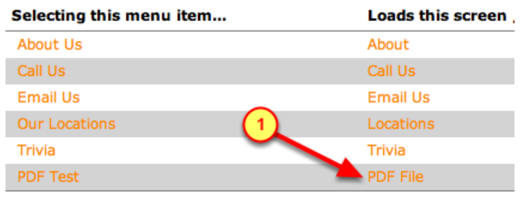
Choose Screen Type: If y drop down list.

NOTE: To add an optional r **"Selecting this menu ite**

Lets get that PDF file into the App by connecting it to a BuzzTouch Menu by adding a new row:

1. Specify the **tile** of the row to show a User
2. Give a **nickname** for the screen that will be displaying the contents of that PDF file
3. The **Screen Type** should be "PDF Doc"
4. Click on the **add** button to create a new row that points to that PDF file

Select the PDF Screen for configuration

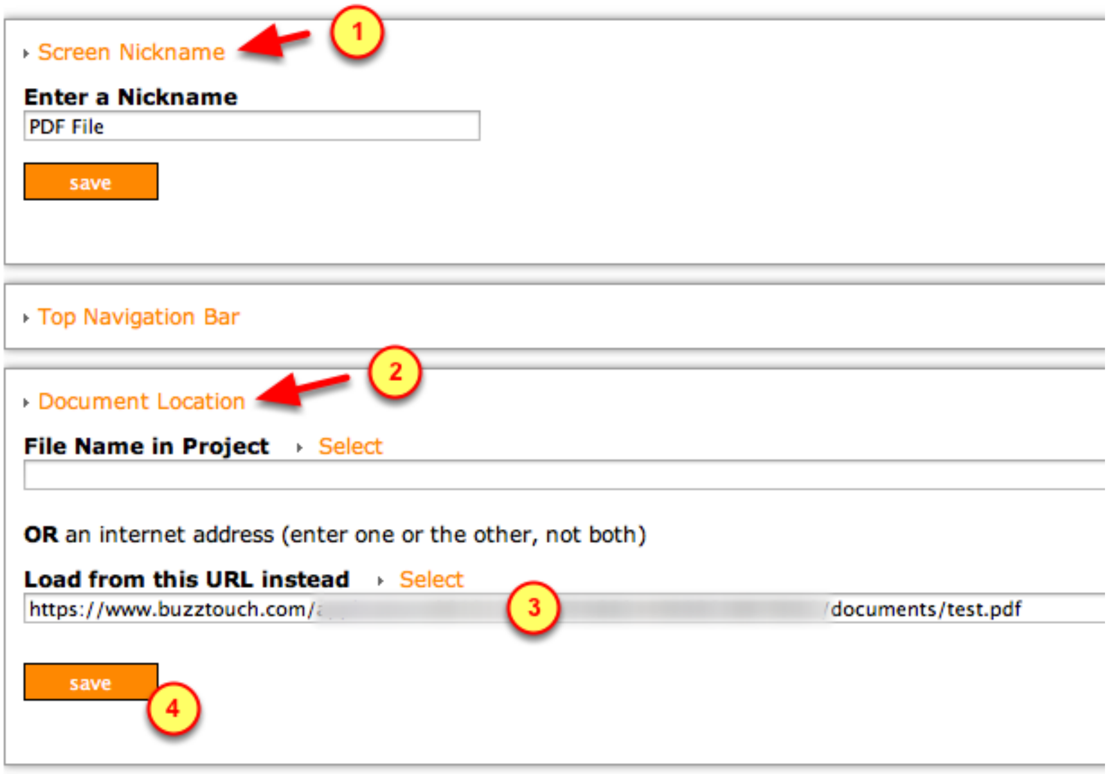


Selecting this menu item...	Loads this screen
About Us	About
Call Us	Call Us
Email Us	Email Us
Our Locations	Locations
Trivia	Trivia
PDF Test	PDF File

Now that we've created a screen to display a PDF, we must connect that screen and the PDF file.

1. Click on the screen for the **PDF File**

Insert link to PDF file


Screen Nickname

Enter a Nickname

PDF File

save

Top Navigation Bar

Document Location

File Name in Project Select

OR an internet address (enter one or the other, not both)

Load from this URL instead Select

https://www.buzztouch.com/[redacted]documents/test.pdf

save

Lets connect the screen that is showing the PDF contents to the actual PDF file

1. Click on **Screen Nickname** to verify we are configuring the correct screen (the one for the PDF File)

2. Click on **Document Location** to fill-in the URL box

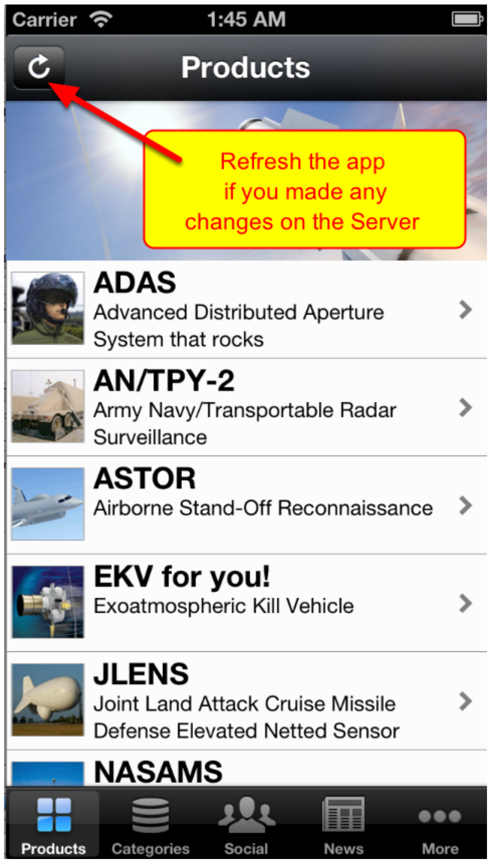
3. Paste in the link to the PDF File into the **URL box**

4. Click on the **save** button

Create a HTML5 screen with custom link

Coming soon ...

Run the App in Simulator



ADAS
Advanced Distributed Aperture System that rocks

AN/TPY-2
Army Navy/Transportable Radar Surveillance

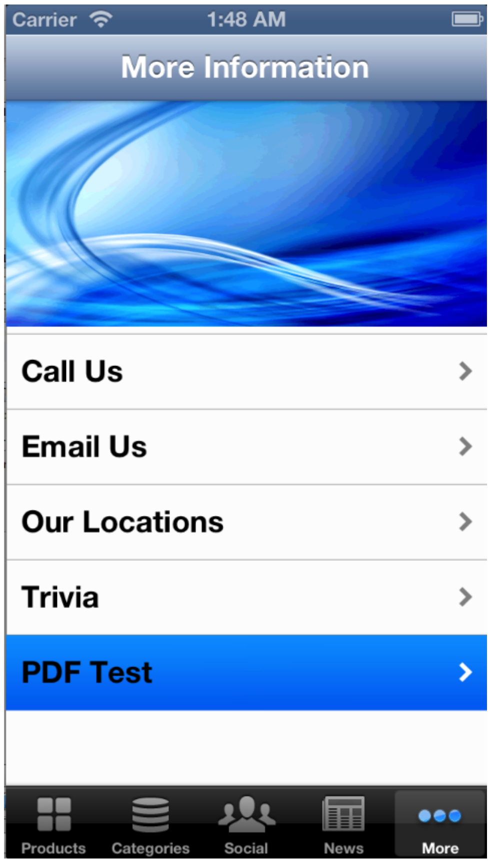
ASTOR
Airborne Stand-Off Reconnaissance

EKV for you!
Exoatmospheric Kill Vehicle

JLENS
Joint Land Attack Cruise Missile Defense Elevated Netted Sensor

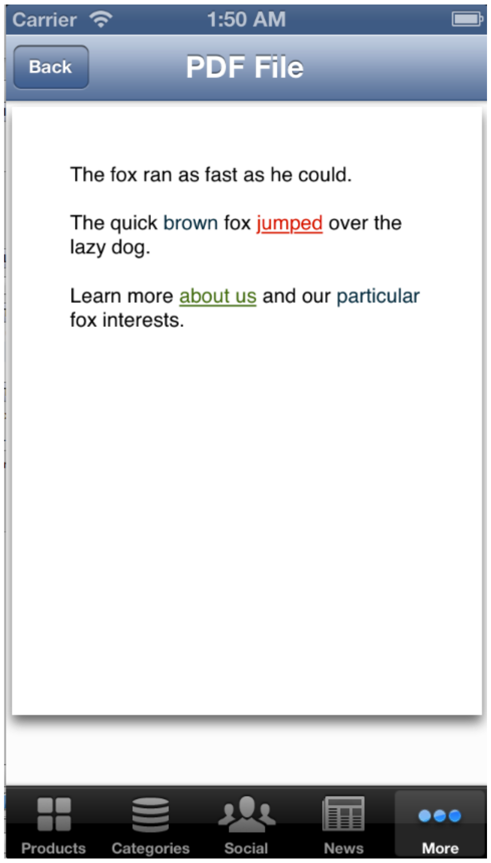
NASAMS

Select the PDF Test menu item



Switch over to the screen that displays the content of the PDF file

Show the contents of the PDF file



The fox ran as fast as he could.

The quick brown fox jumped over the lazy dog.

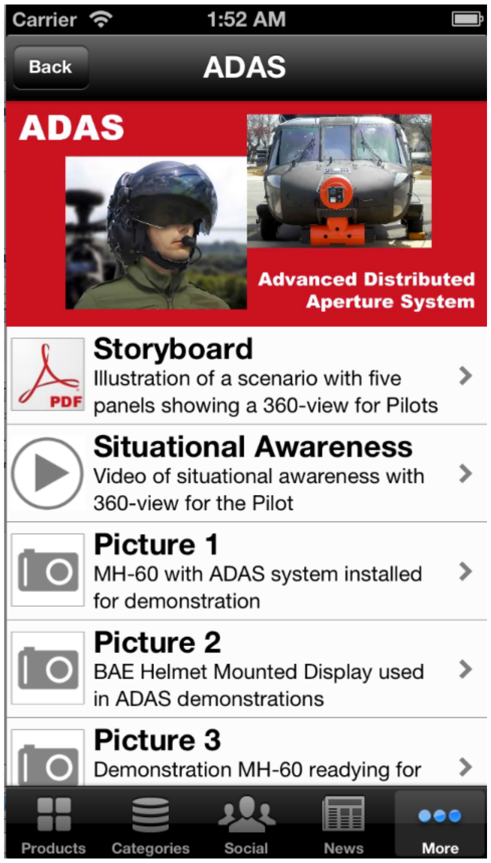
Learn more about us and our particular fox interests.

View the contents of the PDF file. Tap on links that will take you to another screen within the App:

1. Tap on the **jumped** link to see the screen containing the **ADAS product information**

2. Tap on **about us** to read the screen displaying information **about the providers** of this app

Screen on ADAS product information


Carrier 1:52 AM

Back ADAS

Storyboard
Illustration of a scenario with five panels showing a 360-view for Pilots

Situational Awareness
Video of situational awareness with 360-view for the Pilot

Picture 1
MH-60 with ADAS system installed for demonstration

Picture 2
BAE Helmet Mounted Display used in ADAS demonstrations

Picture 3
Demonstration MH-60 readying for

Products Categories Social News More

Screen showing About Us information