Walgreens Prescription API

Health 2.0 Chicago – May 16, 2014

Introduction

The Walgreens Prescription API allows developers to enable their applications to quickly order a Walgreens refill or transfer a refill to one of the 8,000+ Walgreens pharmacies.

The health care focused Prescription API is designed to increase prescription compliance and aid in personal health management through automated refill alerts and a streamlining of the refill process. By offering easy prescription refills through this health management API, Walgreens hopes to further increase health care adherence by reaching a wider audience through third party health care apps.

Prescription Refill

Customers can order refills in seconds, as well as select the option to opt-in to receive a text alert when their prescription order is ready for pick up.

Developers who have scanning functionality within their apps can easily integrate the Prescription API to enable their users to order a prescription refill by scanning the barcode on a Walgreens prescription bottle, similar to the Refill by Scan flow in the Walgreens App.

Prescription Transfer

Customers can transfer their non-Walgreens prescriptions to Walgreens by taking a legible picture of their non-Walgreens prescription, entering some basic information and choosing a Walgreens Store from a list of pick-up store locations per their convenience.

Overview

What this document covers

This document explains how to request information and gain access to the Prescription API.

About the Prescription API

The Prescription API allows developers access to the Walgreens hosted Prescription API mobile checkout flow from within their applications. Applications will be required to pass all required prescription information via a HTTPS POST method.

What this document covers

This document explains how to:

- gain access to the Prescription API
- the overall interactions between an application and Walgreens servers
- use the Prescription API to get the URL to the Walgreens landing page
- handle the transition from a web view back to the application

Typographical conventions

- Methods, variables, field names, and parameter names shall be appear in a fixed-width font
- Blocks of code will appear in gray boxes with a fixed-width font
  - Text with a yellow highlight within a block of code are to be replaced with what the text describes; i.e. if "affiliate ID" is highlighted, then replace that text with an affiliate ID
- Red colored text should always be read carefully

Prerequisites

In order to utilize the API from Walgreens, developers must:

- be developing an application for the U.S
  - Walgreens currently does not have any stores outside of the United States and Puerto Rico
- have a user account on the Walgreens Developer Portal
- have an API key
- have an Affiliate ID
- be developing an application on a platform that supports a browser based on WebKit
  - Refer to "" on page 23 for more details
- have access to a bar code scanner will be only be required if the application will emulate the Refill by Scan feature that appears in the Walgreens apps

Instructions on how to get a user account, API key and Affiliate ID are listed below.

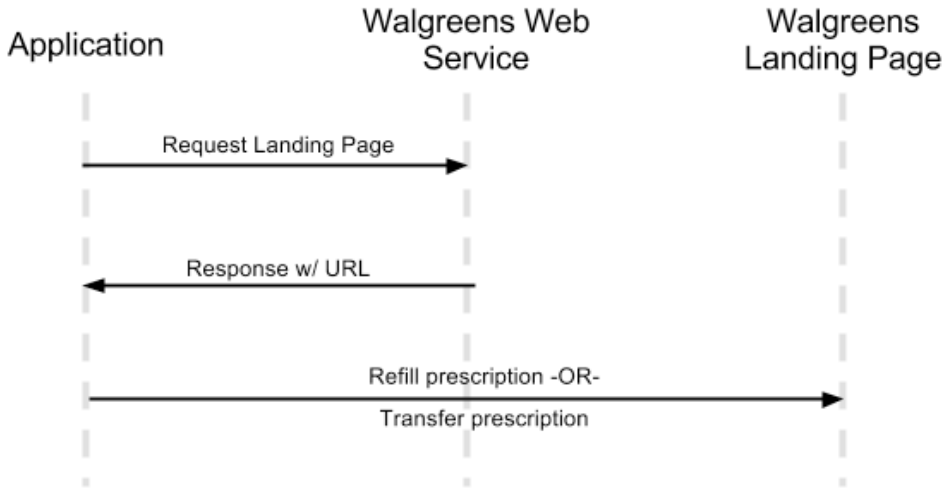
Prescription API Quick Start Guide for Health 2.0 Chicago

Getting Started

- Use the API key value of **7fac7152126efa4c2eabcd33f384eb62**
- Use the Affiliate ID value of **extest1**
- Ensure your API call is pointing to the Walgreens staging environment only (more info on pg 4)
- For completing test prescription refill submissions, utilize any of the numbers below:
  - 0305383-59382
  - 0305384-59382
  - 0305385-59382
  - 0305386-59382

Accessing Prescription API

Overview



The initial access point for the Prescription API flow is via the findURL service, which will provide the URL to the landing page (refer to “” on page 3 for more details). After successfully getting the landing page, applications need to open a web view with the target as the URL to the landing page (a.k.a. landing URL) along with post data. The post data to send to the landing page will vary depending on if the request is for a prescription refill (refer to “” on page 5 for more details) or for a prescription transfer (refer to “” on page 7 for more details).

Note: the information from the findURL service can be reused. So once the landing page information has been successfully received, requests for both refill prescription and transfer prescription can be made using the same landing page information.

Landing Page

The landing URL can be obtained from the findURL web service hosted at:

- https://services-qa.walgreens.com/api/util/mweb5url (staging)
- https://services.walgreens.com/api/util/mweb5url (production)

Request Parameters

Name	Value	Description
`transaction` required	String	Use a value of "`refillByScan`"
`apiKey` required	String	API Key provided by Walgreens
`devinf` required	String	In the comma delimited form of platform then version; refer to "" for examples
`act` required	String	Use a value of "`mweb5Url`"
`view` required	String	Use a value of "`mweb5UrlJSON`"
`affId` required	String	Affiliate ID provided by Walgreens
`appver` required	String	Unique application version number

Sample Request JSON – Staging environment

```
{
     "transaction":"refillByScan",
     "apiKey":"API Key",
     "devinf":"iPhone,6.1",
     "act":"mweb5Url",
     "view":"mweb5UrlJSON",
     "affId":"Affiliate ID",
     "appver":"3.1"
}
```

Sample Response JSON - Staging environment

```
{
     "landingUrl":"https://m5-qa.walgreens.com/mweb5/refillrx/chkExpRx",
     "template":"default",
     "uploadLimit":100,
     "token":"Token",
     "err":""
}
```

Sample Request JSON – Production environment

```
{
     "transaction":"refillByScan",
     "apiKey":"API Key",
     "devinf":"iPhone,6.1",
     "act":"mweb5Url",
     "view":"mweb5UrlJSON",
     "affId":"Affiliate ID",
     "appver":"3.2"
}
```

Sample Response JSON - Production environment

```
{
     "landingUrl":"https://m5.walgreens.com/mweb5/refillrx/chkExpRx",
     "template":"default",
     "uploadLimit":100,
     "token":"Token",
     "err":""
}
```

Note: if the response includes "`uploadLimit`" or "`template`", just ignore those. They are not needed for anything related to prescriptions.

Prescription Refill

Once a landing URL has been obtained, the URL should be loaded in a web view with the request parameters using a HTTPS POST. Refer to "" on page 17 for examples on how to do this for both iOS and Android.

Request Parameters

Name	Value	Description
`rxNo` required	String	Prescription number - the syntax for a Walgreens prescription number is: 7 digits, followed by a dash, followed by 5 digits (e.g. 1234567-12345). The prescription number must be valid in order to move forward with the "Refill By Scan" functionality.
`affId` required	String	Affiliate ID created for each Walgreens partner. An invalid `affId` will throw an exception. An `affID` of "`extest1`" is only for the Staging environment.
`token` required	String	Use the token value sent back in the findURL service
`act` required	String	Use a value of "`chkExpRx`"
`appId` required	String	Use a value of "`refillByScan`"
`appCallBackScheme` required	String	The URL scheme and host used to transfer the control from the web view to the application. Eg: "`appName://handleControlFromScanRefill`", where "`appName`" is the URL scheme and "`handleControlFromScanRefill`" is the host.
`appCallBackAction` required	String	Used to identify the scenario from which the control was returned from the Web view to the native application. Eg: "`callBackAction`"
`devinf` required	String	In the comma delimited form of platform then version
`appver` required	String	Unique application version number
`trackingId` optional	String	Unique value to track a transaction

Sample Request JSON

```
{
     "appId":"refillByScan",
     "affId":"Affiliate ID",
     "token":"Token",
     "rxNo":"Prescription Number",
     "appCallBackScheme":"appName://handleControlFromScanRefill",
     "appCallBackAction":"callBackAction",
     "act":"chkExpRx",
     "trackingId":"Tracking ID",
     "devinf":"iPhone,6.1",
     "appver":"3.2"
}
```

Prescription Transfer

Once a landing URL has been obtained, the URL should be loaded in a web view with the request parameters using a HTTPS POST. Refer to "" on page 17 for examples on how to do this for both iOS and Android. Refer to "" on page 20 for an example image capture implementation for iOS.

Request Parameters

Name	Value	Description
`affId` required	String	Affiliate ID created for each Walgreens partner. An invalid `affId` will throw an exception. An `affID` of "`extest1`" is only for the Staging environment.
`token` required	String	Use the token value sent back in the findURL service
`act` required	String	Use a value of "`transferRxHome`"
`appId` required	String	Use a value of "`transferByScan`"
`appCallBackScheme` required	String	The URL scheme and host used to transfer the control from the web view to the application. Eg: "`appName://handleControlFromScanRefill`", where "`appName`" is the URL scheme and "`handleControlFromScanRefill`" is the host.
`appCallBackAction` required	String	Used to identify the scenario from which the control was returned from the Web view to the native application. Eg: "`callBackAction`"
`devinf` required	String	In the comma delimited form of platform then version
`appver` required	String	Unique application version number
`rxImg` required	String	Image data as a Base64 string
`trackingId` optional	String	Unique value to track a transaction
`fname` optional	String	Customer's first name
`lname` optional	String	Customer's last name
`dob` optional	String	Customer's date of birth; the string must be in the following format: MM-DD-YYYY For example of April 2, 1998: "`04-02-1998`"
`pharmacyNbr` optional	String	Prescription's original pharmacy's phone number
`phoneNbr` optional	String	Customer's phone number

Note:

- Submit the following optional fields to prepopulated their corresponding fields in the web view:
  - fname
  - lname
  - dob
  - pharmacyNbr
  - phoneNbr
- Consider re-compressing the image of the prescription before converting it to a Base64 string, so the user will upload a smaller file, which will speed up the overall response time.

Example of re-compressing for iOS

```
// A compression quality of 0.2 has been tested and works well
NSData *imgData = UIImageJPEGRepresentation(imgRx, 0.2);
```

Sample Request JSON

```
{
     "appId":"transferByScan",
     "affId":"Affiliate ID",
     "token":"Token",
     "appCallBackScheme":"appName://handleControlFromScanRefill",
     "appCallBackAction":"callBackAction",
     "act":"transferRxHome",
     "devinf":"iPhone,6.1",
     "rxImg": "Image data as a Base64 string",
     "appver":"3.2",
     "trackingId":"Tracking ID",
     "fname":"First name",
     "lname":"Last name",
     "dob":"MM-DD-YYY",
     "pharmacyNbr":"Pharmacy's phone number",
     "phoneNbr":"Customer's phone"
}
```

Handing Back Control to the Application

The user can be transferred from the Web view to the application by:

- 5 possible ways when refilling a prescription:
  - When the user taps the "Cancel" button on the landing page
  - When the user taps the "Back" button on the landing page
  - When the user taps the "Close" button on the prescription confirmation page
  - When the user taps the "Refill Another" button on the prescription confirmation page
  - After an invalid prescription number alert
- 6 possible ways when transferring a prescription
  - When the user taps the "Cancel" button on the landing page
  - When the user taps the "Back" button on the landing page
  - When the user taps the "Cancel" button on the transfer review page
  - When the user taps the "Home" button on the transfer confirmation page
  - When the user taps the "Transfer Another" button on the transfer confirmation page
  - When the user taps the "Done" button on the transfer confirmation page

In the web view, when one of the buttons mentioned above is selected, a call will be made with the following syntax:

> Value of `appCallBackScheme` passed in the request + "?" + value of `appCallBackAction` passed in the request + "=" + the callback value

Callback Values for Prescription Refills

Button Pressed by User	Callback Value
Cancel	"cancel"
Back	"back"
Close	"close"
Refill Another	"fillAnother"
Ok (in an invalid prescription number alert)	"refillTryAgain"

Callback Values for Prescription Transfer

Page	Button Pressed	Callback Value
Landing	Back	"txCancel"
Landing	Cancel	"txCancel"
Transfer Review	Cancel	"txCancel"
Transfer Confirmation	Home	"txHome"
Transfer Confirmation	Transfer Another	"txAnother"
Transfer Confirmation	Done	"txDone"

Example of a call from the "refill another" button being pressed

```
refill://handle?callBackAction=refillAnotherRx
```

In the example above, the application's request had:

- `appCallBackScheme` set to "`refill://handle`"
- `appCallBackAction` set to "`callBackAction`"

The way the application handles this call is dependent on the operating system the application is running on. The following sections provide information on how an application can handle the transition for:

-
-

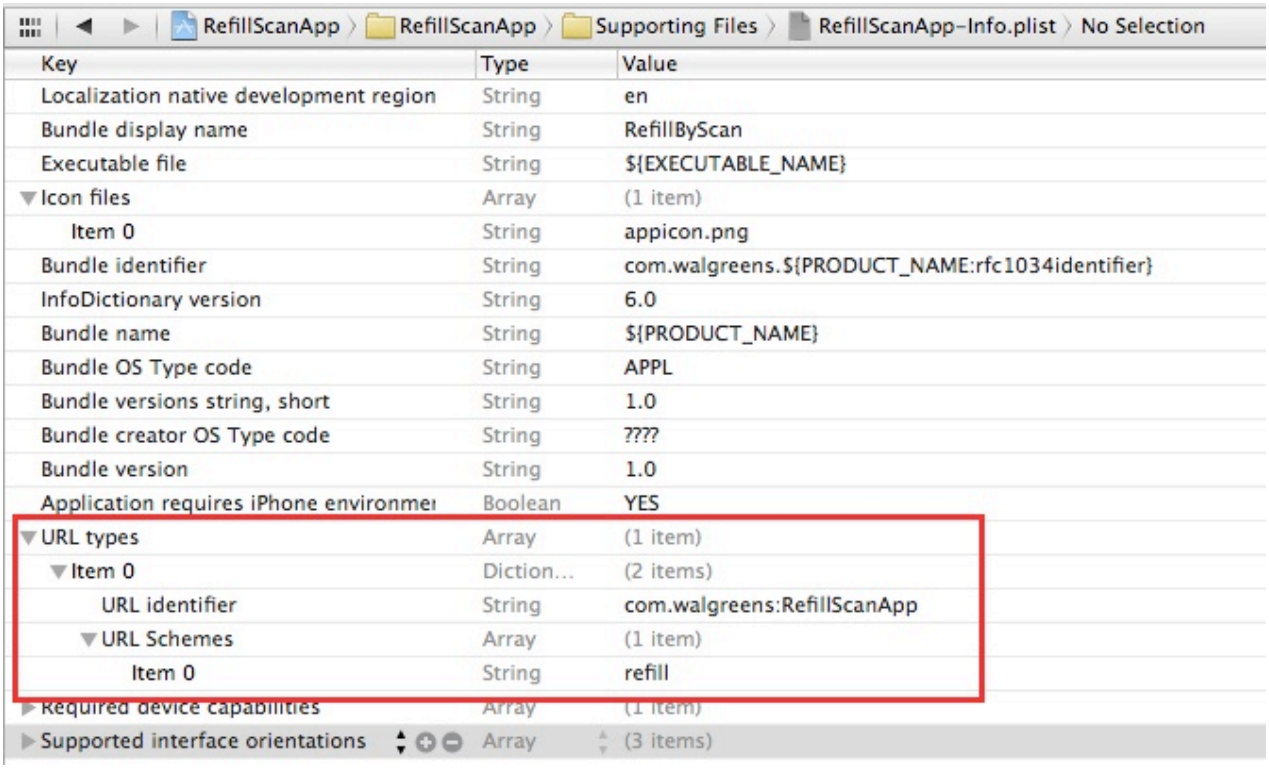
Note: the information provided is not the only way to handle the transition; it's just one way that's easy to port across platforms.

iOS

In an iOS application, the following need to be accounted for, in order to handle the transition:

- Add the URL identifier and the URL schemes to the application's info.plist
- Handle the control based on the query string returned from the Web view

Modifying info.plist



Open info.plist in Xcode, then add in a row for "URL types". Expand the item in the new row, then enter in the URL identifier. Add another new row to the item that was expanded for "URL Schemes", then enter in the URL scheme.

IMPORTANT
Make sure the URL scheme used in the plist matches the URL scheme portion of the value used in `appCallBackScheme`

Transition from the web view to the app

Once one of the buttons mentioned in “” is selected, the control from the web view to the application is transferred via the AppDelegate’s callback method.

Update the following function in AppDelgate.m to handle all the transitions:

```
-(BOOL)application:(UIApplication *)application openURL:(NSURL *)url
     sourceApplication:(NSString *)sourceApplication
     annotation:(id)annotation {
     if([[url scheme] isEqualToString:@"refill"]) {
           // Implementation logic goes here
           // Note: @"refill" is from the URL scheme in the plist file
     }
     return YES;
}
```

Android

In an Android application, the following need to be accounted for, in order to handle the transition:

- Manifest must be updated
- Handle the control based on the query string returned from the Web view

Modifying the Manifest

The Android application's manifest file, AndroidManifest.xml, must be updated with the following changes:

```
<activity android:label="@string/app_name" android:name=".DataActivity" >
  <intent-filter>
     <action android:name="android.intent.action.VIEW"></action>
     <category android:name="android.intent.category.DEFAULT"></category>
     <category android:name="android.intent.category.BROWSABLE"></category>
     <data android:scheme="refillByScan" android:host="handleControl"></data>
  </intent-filter>
</activity>
```

IMPORTANT
Make sure the URL scheme used in the manifest matches the URL scheme portion of the value used in `appCallBackScheme`

Transition from the web view to the app

DataActivity is the class where the control is returned when any of the buttons in “” is pressed from the web view. DataActivity needs to be updated to handle control from the web view:

```
public class DataActivity extends DroidGap {
     @Override
     public void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           Uri url = getIntent().getData();
           if(url != null && url.getScheme().equalsIgnoreCase("refillByScan")){
                 String action = url.getQueryParameter("callBackAction");
                 Intent newIntent = new Intent();
                 if (action.equalsIgnoreCase("back")) {
                       // Implementation Logic goes here
                 } else if (action.equalsIgnoreCase("cancel")) {
                       // Implementation Logic goes here
                 } else if (action.equalsIgnoreCase("close")) {
                       // Implementation Logic goes here
                 } else if (action.equalsIgnoreCase("fillAnother")) {
                       // Implementation Logic goes here
                 } else if (action.equalsIgnoreCase("refillTryAgain")) {
                       // Implementation Logic goes here
                 }
           }
     }

     // IMPORTANT:
     // DroidGap class's onKeyUp() must be overridden.
     // Note: Override onKeyUp() to implement Android's native back button
     //       functionality
     @Override
     public boolean onKeyUp(int keyCode, KeyEvent event) {
           if (keyCode == KeyEvent.KEYCODE_BACK) {
                 //call the method defined in javascript
                 this.loadUrl("javascript:nativeBackBtnClicked();");
                 return true;
           }
           return false;
     }
}
```

Appendix

Error Codes & Messages for Prescription Refills

API Server Side Error Codes & Messages

ERROR CODE	ERROR MESSAGE
111,112,113,114	Configuration issues found please check configuration
115	There is an exception that occurred. Invalid Token.
116	Action or Service handler mapping is wrong. Please check Configuration file.
117	JSON request is incorrect
118	A required attribute (either Affiliate ID, App ID, or Action ID) is missing in the request
123	The prescription number you entered cannot be filled using this feature. You will need to place your order using our "Refill from Account" option.
124	Please enter a valid prescription number to continue
125	Unable to refill your prescription now. Please contact the Pharmacist.
126	Sorry, network error; please try again later
128	Network error, please try again
1001	Sorry, failed to subscribe for text messages
202	Sorry, Service error; please try again later
205	No request parameters found
209	No stores found based on your current location
ERR_DEFAULT	Internal System error occurred. Please contact the Administrator
ERR_PRES_CANT_PICKED	Unable to refill your prescription now. Please contact the Pharmacist.
VALIDATE_RX_WRN_MSG	There are no refills left for this prescription
ERR_TEXT_SUBSCR	Sorry, failed to subscribe for text messages
ERR_NET_FAIL	Sorry, service error. Please try again later.
ERR_APP_ID_INCORRECT	Requested application not found in the configuration file

HTML5 Checkout UI Error Codes & Messages

ERROR CODE	ERROR MESSAGE
ALERT_ERR_INVALID_TIME	Please select valid store time
ALERT_FINDSTR_SEARCH_FIELD_EMPTY	Please enter valid State, City, or Zip Code
ALERT_NO_STORES_FOUND	No stores found based on your current location
ALERT_TOGGLE_SWITCH_ON	Please toggle ON to subscribe for text messages
ALERT_ERR_VALID_PHNO	Please enter a valid phone number

Error Codes & Messages for Prescription Transfers

API Server Side Error Codes & Messages

ERROR CODE	ERROR MESSAGE
632	Mandatory field is empty
635	Sorry, failed to subscribe for text messages
636	Invalid Pick-up Time
633	Invalid Pharmacy Phone Number
630	Invalid Phone Number
658	Missing device info
603	Error generating PDF
638	Invalid SMS OPT Service
111,112,113,114	Configuration issues found please check configuration
115	There is an exception that occurred. Invalid Token
116	Action or Service handler mapping is wrong Pls check Configuration file
117	JSON request is incorrect
118	One or more required attributes (Affiliate ID, App ID, or Action ID) are missing in the request
202	Sorry, Service error. Please try again later
205	No request parameters found
209	No stores found based on your current location

HTML5 UI Error codes & Messages

ERROR CODE	ERROR MESSAGE
ERR_GEN_PDF	Network issues are preventing the transfer. Please try again later.
ERR_MISSING_MANDATORY_FLDS	Please enter all required information
ERR_INVALID_PH_NO	Please enter a valid phone number
ERR_INVALID_PHARMACY_PH_NO	Please enter a valid pharmacy phone number
ERR_NO_RESPONSE	Network error; no response from service
ERR_DEFAULT	Internal System error occurred. Please contact the Administrator.
ALERT_FINDSTR_SEARCH_FIELD_EMPTY	Please enter a valid State, City, or Zip Code
ALERT_NO_STORES_FOUND	No stores found based on the current location
ERR_APP_ID_INCORRECT	Requested application not found in the configuration file

Generating a Request to Refill a Prescription

Once the Landing URL has been successfully retrieved from the findURL web service, post the request JSON parameters to the Landing URL in the UIWebView.

Example of a method for iOS that posts the request to the API through the UIWebView

```
-(void)invokeWebApp {
    // dictResponseValues is a dictionary that contains the first response
    // mwebview is a UIWebView

    NSMutableDictionary *dictRequest = [[NSMutableDictionary alloc] init];

    [dictRequest setValue:@"chkExpRx" forKey:@"act"];
    [dictRequest setValue:@"refillByScan" forKey:@"appId"];
    [dictRequest setValue:[dictResponseValues valueForKey:@"token"]
                  forKey:@"token"];
    [dictRequest setValue:@"extest1" forKey:@"affId"];
    [dictRequest setValue:@"1234567-12345" forKey:@"rxNo"];
    [dictRequest setValue:@"refill://handle" forKey:@"appCallBackScheme"];
    [dictRequest setValue:@"callBackAction" forKey:@"appCallBackAction"];
    [dictRequest setValue:@"iPhone,6.1" forKey:@"devinf"];
    [dictRequest setValue:@"3.3.1" forKey:@"appver"];

    NSString *reqStr = [dictRequest JSONRepresentation];
    NSURL *url = [NSURL URLWithString:
                      [dictResponseValues valueForKey:@"landingUrl"]];
    NSMutableURLRequest *request =
                      [[NSMutableURLRequest alloc]initWithURL:url];
    [request setHTTPMethod:@"POST"];
    [request setHTTPBody:[reqStr dataUsingEncoding:NSUTF8StringEncoding]];

    [mwebview loadRequest:request];
}
```

Example of a method for Android that posts the request to the API through the UIWebView

```
public void postRequest() {
     // webview is a web view

     String webRes = getWebserviceResponse();
     if(webRes != null) {
           Map<String,String> webResMap = gson.fromJson(webRes, Map.class);
           String strLandingUrl = webResMap.get("landingUrl");
           String token = webResMap.get("token");
           MWebRequest requestObj = new MWebRequest();
           requestObj.setAct("chkExpRx");
           requestObj.setAppId("refillByScan");
           requestObj.setAffId("extest1");
           requestObj.setRxNo("1234567-12345");
           requestObj.setToken(token);
           requestObj.setAppCallBackScheme("refillByScan://handleControl");
           requestObj.setAppCallBackAction("callBackAction");
           requestObj.setDevInf("Android,2.3.3");
           requestObj.setAppver("3.3");
           String gsonStr = gson.toJson(requestObj);
           byte[] postData = EncodingUtils.getBytes(gsonStr, "BASE64");
           webview.postUrl(strLandingUrl, postData);
}
```

Example of Image Capture on iOS

The following code example shows how to:

- invoke the native camera with video mode disabled
- invoke a help screen
- capture an image

Example code for iOS to invoke the native camera

```
-(void) invokeCameraForTransfer {
     imgPicker = [[UIImagePickerController alloc] init];
     imgPicker.sourceType = UIImagePickerControllerSourceTypeCamera;
     imgPicker.cameraCaptureMode=UIImagePickerControllerCameraCaptureModePhoto;
     imgPicker.allowsEditing = NO;
     imgPicker.delegate = self;

     // Navigation bar implementation
     UINavigationBar *navBar = [[UINavigationBar alloc]
           initWithFrame:CGRectMake(0, 0, 320, 44)];
     [navBar setTintColor:[UIColor colorWithRed:180.0
                                          green:0.0
                                           blue:0.0
                                          alpha:1.0]];
     UIButton *btn = [UIButton buttonWithType:UIButtonTypeInfoLight];

     // Help screen overlay should be invoked on info button click
     UIBarButtonItem *infoButton = [[UIBarButtonItem alloc]
           initWithCustomView:btn];
     [btn addTarget:self action:@selector(invokeInfoScreen)
               forControlEvents:UIControlEventTouchUpInside];

     // Add button to the navigation bar
     UINavigationItem *item = [[UINavigationItem alloc]
           initWithTitle:@"Transfer Rx"];
     item.rightBarButtonItem = infoButton;
     item.hidesBackButton = YES;
     [navBar pushNavigationItem:item animated:YES];
     [infoButton release];
     [item release];
     [imgPicker.view addSubview:navBar];

     // Information Overlay
     UILabel *lblInfo = [[UILabel alloc]
           initWithFrame:CGRectMake(20, 90, 280, 60)];
     [lblInfo setNumberOfLines:3];
     [lblInfo setTextAlignment:NSTextAlignmentCenter];
     [lblInfo setText:@"Transfer from other pharmacy to Walgreens. Position
bottle in screen so prescription number and pharmacy name are visible."];
     [lblInfo setTextColor:[UIColor whiteColor]];
     [lblInfo setBackgroundColor:[UIColor clearColor]];
     [lblInfo setFont:[UIFont fontWithName:@"Trebuchet MS" size:12.0f]];
     [imgPicker.view addSubview:lblInfo];
     [self presentViewController:imgPicker animated:YES completion:nil];
}
```

Example code for iOS for the help screen

```
-(void)invokeInfoScreen {
     // help screen view
     infoView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];
     [infoView setBackgroundColor:[UIColor blackColor]];

     // help screen image view
     UIImageView *imgView =[[UIImageView alloc]
           initWithFrame:CGRectMake(0, 0, 320, 480)];
     [infoView addSubview:imgView];

     // close button for the help screen (overlay)
     UIButton *btnInInfoScreen = [[UIButton alloc]
           initWithFrame:CGRectMake(100, 100, 60, 30)];
     [btnInInfoScreen setBackgroundColor:[UIColor whiteColor]];
     [btnInInfoScreen setTitle:@"Close" forState:UIControlStateNormal];
     [btnInInfoScreen setTitleColor:[UIColor redColor]
                           forState:UIControlStateNormal];
     [btnInInfoScreen addTarget:self
                         action:@selector(closeInfoScreen)
               forControlEvents:UIControlEventTouchUpInside];
     [infoView addSubview:btnInInfoScreen];
     [imgPicker.view addSubview:infoView];
}

// method to close the help screen
-(void) closeInfoScreen {
    [infoView removeFromSuperview];
}
```

Example code for iOS for the delegate methods for the camera

```
// method to capture the image
- (void) imagePickerController: (UIImagePickerController*) ireader
     didFinishPickingMediaWithInfo: (NSDictionary*) info {
     self.imgRx = [info objectForKey:UIImagePickerControllerOriginalImage];
     [ireader dismissViewControllerAnimated:YES completion:nil];
     [self performSelector:@selector(invokeWebView)
                withObject:nil
                afterDelay:0.5];
}

- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {
     [picker dismissViewControllerAnimated:YES completion:nil];
}
```

Examples Values for devinfo

Examples of the Device Information parameter

Example Device Type	Example Platforms	Example Values for the parameter `devinf`
iPhone	6.1	`"iPhone,6.1"`
iPad	6.0	`"iPad,6.0"`
iPod	6.0	`"iPod,6.0"`
Android	2.3.3	`"Android,2.3.3"`
Android – Tablet	3.2	`"AndroidTablet,3.2"`
Blackberry	7.1	`"Blackberry,7.1"`
Blackberry – Playbook	1.0	`"BlackberryPlaybook,1.0"`
Windows	6.0	`"Windows,6.0"`

Supported Mobile Operating System Information

The Prescription API mobile checkout pages are constructed utilizing jQuery Mobile. For optimal performance of these pages within your application, the following operating systems should be utilized:

- Apple iOS – version 3.2 and higher
- Android – version 2.3 and higher
- Windows Phone – version 7.5 and higher
- BlackBerry – version 6.0 and higher
- BlackBerry Playbook – version 1.0 and higher

For additional information regarding jQuery mobile's browser support, please visit http://jquerymobile.com/gbs.