Invoking adapter procedures from native iOS applications

To create and configure an iOS native project, first follow the "Creating your first Native iOS MobileFirst application (../../hello-world/creating-first-native-ios-mobilefirst-application/)" tutorial .

Initializing WLClient

1. Access the `WLClient` functionality by using `[WLClient sharedInstance]` anywhere in your application.
2. Initiate the connection to the server by using the `wlConnectWithDelegate` method. For most actions, you must specify a delegate object, such as a `MyConnectListener` instance in the following example:

   ```
   MyConnectListener *connectListener = [[MyConnectListener alloc] initWithController:self];
   [[WLClient sharedInstance] wlConnectWithDelegate:connectListener];
   ```

   Note: Remember to import **WLClient.h** and **WLDelegate.h** in your header file.

   You must supply a connection delegate (listener) to the MobileFirst invocation methods.
3. Create a delegate to be used in the `wlConnectWithDelegate` method and receive the response from the MobileFirst Server. Name the class `MyConnectListener`. The header file must specify that it implements the `WLDelegate` protocol.

   ```
   @interface MyConnectListener : NSObject <WLDelegate> {
      @private
      ViewController *vc;
   }
   ```

   The `WLDelegate` protocol specifies that the class implements the following methods: - The **onSuccess** method: `(WLResponse *)response` - The **onFailure** method: `(WLFailResponse *)response`
   After `wlConnectWithDelegate` finishes, the `onSuccess` method or the `onFailure` method of the supplied `MyConnectListener` instance is invoked. In both cases, the response object is sent as an argument.
4. Use this object to operate data that is retrieved from the server.

   ```
   -(void)onSuccess:(WLResponse *)response{
      NSLog(@"\nConnection Success: %@", response);
      NSString *resultText = @"Connection success. ";
      if ([response responseText] != nil){
         resultText = [resultText stringByAppendingString:[response responseText]]
   ;
      }
      [vc updateView:resultText];
   }
   -(void)onFailure:(WLFailResponse *)response{
      NSString *resultText = @"Connection failure. ";
      if ([response responseText] != nil){
         resultText = [resultText stringByAppendingString:[response responseText]]
   ;
      }
      [vc updateView:resultText];
   }
   ```

Invoking an adapter procedure

To invoke a procedure, create a `WLProcedureInvocationData` object and specify the adapter name and the procedure name. Invoke the procedure by using the shared instance of the `WLClient`.

```
    WLProcedureInvocationData *myInvocationData = [[WLProcedureInvocationData alloc] initWithAdapterName:@"RS
SReader" procedureName:@"getStories"];
    MyInvokeListener *invokeListener = [[MyInvokeListener alloc] initWithController: self];
    [[WLClient sharedInstance] invokeProcedure:myInvocationData withDelegate:invokeListener];
```

As previously stated, you must supply a delegate object to manage the retrieved data.

Receiving a procedure response

When the procedure invocation is complete, a delegate method of `MyInvokeListener` class instance is called. Any delegate header file must specify that it complies with a `WLDelegate` protocol.

```
@interface MyInvokeListener : NSObject <WLDelegate> {
@private
    ViewController *vc;
    NSString *strResponse;
}
- (id)initWithController: (ViewController *) mainView;</p>
@end
```

After the procedure invocation finishes, the `onSuccess` method or the `onFailure` method of the supplied `MyInvokeListener` instance is called. In both cases, a response object is sent as an argument. Use this object to operate data that is retrieved from the server.

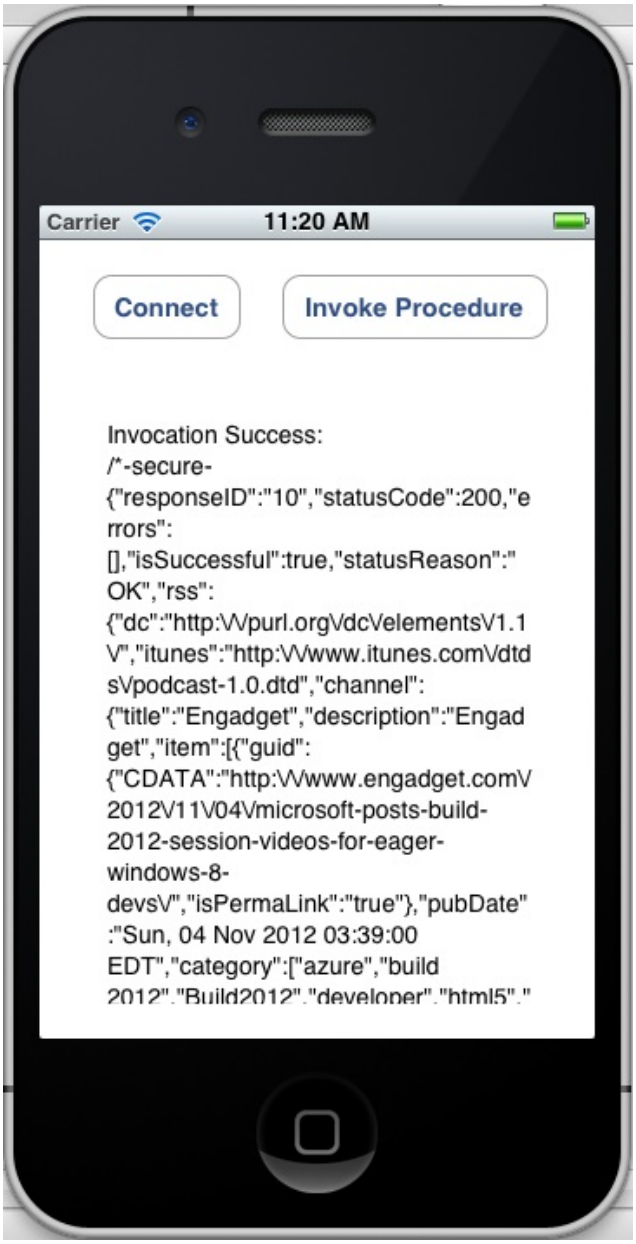
```
-(void)onSuccess:(WLResponse *)response {
    NSLog(@"Invocation Success: %@", response);
    NSString *resultText = @"Invocation success. ";</p>
   if ([response responseText] != nil){
        resultText = [resultText stringByAppendingString:[response responseText]]
;
    }
    [vc updateView:resultText];
}
-(void)onFailure:(WLFailResponse *)response{
    NSLog(@"Invocation Failure: %@", response);
    NSString *resultText = @"Invocation failure. ";</p>
    if ([response responseText] != nil){
        resultText = [resultText stringByAppendingString:[response responseText]]
;
    }
    [vc updateView:resultText];
}
```

Sample application

The sample contains two projects: - The **InvokingAdapterProceduresNativeProject.zip** file contains a MobileFirst native API that you can deploy to your MobileFirst server. - The **InvokingAdapterProceduresiOSProject.zip** file contains a native iOS application that uses a MobileFirst native API library to communicate with the MobileFirst Server.

Make sure to update the **worklight.plist** file in **iOSNativeApp** with the relevant server settings.

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/InvokingAdapterProceduresNativeProject.zip) the Studio project. Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/InvokingAdapterProceduresiOSProject.zip) the Native project.


Carrier
11:20 AM
Connect
Invoke Procedure
Invocation Success:
/*-secure-
{"responseID":"10","statusCode":200,"errors":[],"isSuccessful":true,"statusReason":"OK","rss":{"dc":"http:\/\/purl.org\/dc\/elements\/1.1\/","itunes":"http:\/\/www.itunes.com\/dtds\/podcast-1.0.dtd","channel":{"title":"Engadget","description":"Engadget","item":[{"guid":{"CDATA":"http:\/\/www.engadget.com\/2012\/11\/04\/microsoft-posts-build-2012-session-videos-for-eager-windows-8-devs\/","isPermaLink":"true"},"pubDate":"Sun, 04 Nov 2012 03:39:00 EDT","category":["azure","build 2012","Build2012","developer","html5",