Additional Information

Working with bitcode in iOS apps

- The application-authenticity security check is not supported with bitcode.
- watchOS applications require bitcode enabled.

To enable bitcode, in your Xcode project navigate to the **Build Settings** tab and set **Enable Bitcode** to **Yes**.

Enforcing TLS-secure connections in iOS apps

Starting from iOS 9, Transport Layer Security (TLS) protocol version 1.2 must be enforced in all apps. You can disable this protocol and bypass the iOS 9 requirement for development purposes.

Apple App Transport Security (ATS) is a new feature of iOS 9 that enforces best practices for connections between the app and the server. By default, this feature enforces some connection requirements that improve security. These include client-side HTTPS requests and server-side certificates and connection ciphers that conform to Transport Layer Security (TLS) version 1.2 using forward secrecy.

For **development purposes**, you can override the default behavior by specifying an exception in the info.plist file in your app, as described in App Transport Security Technote. However, in a **full production** environment, all iOS apps must enforce TLS-secure connections for them to work properly.

To enable non-TLS connections, the following exception must appear in the **project-name-info.plist** file in the **project-name\Resources** folder:

```
<key>NSExceptionDomains</key>
  <dict>
    <key>yourserver.com</key>

      <dict>
      <!--Include to allow subdomains-->
      <key>NSIncludesSubdomains</key>
      <true/>

      <!--Include to allow insecure HTTP requests-->
      <key>NSTemporaryExceptionAllowsInsecureHTTPLoads</key>
      <true/>
    </dict>
  </dict>
```

To prepare for production

1. Remove, or comment out the code that appears earlier in this page.
2. Set up the client to send HTTPS requests by using the following entry to the dictionary:

   ```
   <key>protocol</key>
   <string>https</string>

   <key>port</key>
   <string>10443</string>
   ```

   The SSL port number is defined on the server in **server.xml** in the `httpEndpoint` definition.
3. Configure a server that is enabled for the TLS 1.2 protocol. For more information, see Configuring MobileFirst Server to enable TLS V1.2 (http://www-01.ibm.com/support/docview.wss?uid=swg21965659)

4. Make settings for ciphers and certificates, as they apply to your setup. For more information, see App Transport Security Technote (https://developer.apple.com/library/prerelease/ios/technotes/App-Transport-Security-Technote/), Secure communications using Secure Sockets Layer (SSL) for WebSphere Application Server Network Deployment (http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/csec_sslsecurecom.html?cp=SSAW57_8.5.5%2F1-8-2-33-4-0&lang=en), and Enabling SSL communication for the Liberty profile (http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.doc/ae/twlp_sec_ssl.html?cp=SSAW57_8.5.5%2F1-3-11-0-4-1-0).

Enabling OpenSSL for iOS

The MobileFirst iOS SDK uses native iOS APIs for cryptography. You can configure the IBM MobileFirst™ Platform Foundation V8.0.0 to use the OpenSSL cryptography library in iOS apps.

Encryption/decryption is provided with the following APIs: `WLSecurityUtils.encryptText()` and `WLSecurityUtils.decryptWithKey()`.

Option 1: Native encryption and decryption

Native encryption and decryption is provided by default, without using OpenSSL. This is equivalent to explicitly setting the encryption or decryption behavior as follows:

```
WLSecurityUtils enableOSNativeEncryption:YES
```

Option 2: Enabling OpenSSL

OpenSSL is disable by default. To enable it, proceed as follows:

1. Install the OpenSSL frameworks:
   - With CocoaPods: Install the `IBMMobileFirstPlatformFoundationOpenSSLUtils` pod with CocoaPods.
   - Manually in Xcode: Link the `IBMMobileFirstPlatformFoundationOpenSSLUtils` and openssl frameworks manually in the Link Binary With Libraries section of the Build Phases tab.
2. The following code enables the OpenSSL option for the encryption/decryption:

   ```
   WLSecurityUtils enableOSNativeEncryption:NO
   ```

   The code will now use the OpenSSL implementation if found and otherwise throw an error if the frameworks are not installed correctly.

With this setup, the encryption/decryption calls use OpenSSL as in previous versions of the product.

Migration options

If you have an MobileFirst project that was written in an earlier version, you might need to incorporate changes to continue using OpenSSL. * If the application is not using encryption/decryption APIs and no encrypted data is cached on the device, no action is needed. * If the application is using encryption/decryption APIs, you have the option of using these APIs with or without OpenSSL.

Migrating to native encryption

1. Make sure the default native encryption/decryption option is chosen (see Option 1).
2. Migrating cached data: If the previous installation of IBM MobileFirst Platform Foundation saved encrypted data to the device using OpenSSL, OpenSSL frameworks must be installed as described in Option 2. The first time the application attempts to decrypt the data it will fall back to OpenSSL and then encrypt it using native encryption. If the OpenSSL framework is not installed an error is thrown. This way the data will be auto-migrated to native encryption allowing subsequent releases to work without the OpenSSL framework.

Continuing with OpenSSL

If OpenSSL is required use the setup described in Option 2.

Last modified on