Additional Information

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 d**evelopment 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 in Cordova Applications

The MobileFirst Cordova SDK for iOS uses native iOS APIs for cryptography. You can configure the application to instead use the OpenSSL cryptography library in your Cordova iOS app.

The encryption/decryption functionalities are provided with the following Javascript APIs:

- WL.SecurityUtils.encryptText
- WL.SecurityUtils.decryptWithKey

Option 1: Native encryption/decryption

By default MobileFirst provides native encryption/decryption, without using OpenSSL. This is equivalent to explicitly setting the encryption/decryption behavior:

- WL.SecurityUtils.enableNativeEncryption(true)

Option 2: Enabling OpenSSL

MobileFirst provided OpenSSL is disabled by default.

To install the necessary frameworks for supporting OpenSSL, first install the Cordova plug-in:

```
cordova plugin add cordova-plugin-mfp-encrypt-utils
```

The following code enables the OpenSSL option for the encryption/decryption:

- WL.SecurityUtils.enableNativeEncryption(false)

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

Migration options

If you have an IBM MobileFirst Foundation project written in an earlier version, you may 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, but the native encryption/decryption option is now chosen, the stored data must be decrypted. The first time the application attempts to decrypt the data it will fall back to OpenSSL and then encrypt it using native encryption. This way the data will be auto-migrated to native encryption. **Note:** To allow the decryption from OpenSSL, you must add the OpenSSL frameworks by installing the Cordova plug-in: `cordova plugin add cordova-plugin-mfp-encrypt-utils`
    - **Continuing with OpenSSL:** If OpenSSL is required use the setup described in **Option 2**.

Last modified on