TEMENOS
The Banking Software Company

Guide for Developers

App2App message relay plugin

Information in this document is subject to change without notice.

No part of this document may be reproduced or transmitted in any form or by any means, for any purpose, without the express written permission of TEMENOS HEADQUARTERS SA.

COPYRIGHT © 2007 - 2014 TEMENOS HEADQUARTERS SA. All rights reserved.

Table of Contents

Document History

Author	Version	Date	Details
Horia Popescu	1.0	15.12.2017	Creation
Horia Popescu	1.1	15.12.2017	Added plugin details provided by Arun
Horia Popescu	1.2	12.01.2018	Added details about patching an existing project using the new components
Horia Popescu	1.3	21.05.2018	Updated to include the Android part of the plugin
Horia Popescu	1.4	05.06.2018	Updated images to reflect the name change
Arun Vamadevan	1.5	26.06.2018	Plugin configurations are updated

Comments:

Version 1.3 also comes with a change in naming of the plugin from InAppBrowserWKWebView to App2app message relay plugin

1 Introduction

The purpose of this document is to provide a description of what has been done for the new App2app feature required by Raiffeisen. This document assumes that work is being done on the TCIB Retail UXP project which was customized for Raiffeisen. There will also be information on how to use the components presented here in a standalone manner.

This is intended for developers and people that will need to use this feature.

2 Overview

What was developed is a way to allow the bank customer to use the LuxTrust Mobile app as part of the login flow. The LuxTrust Mobile app will act as an OTP generator. An overview of the login flow:

1. The user enters Retail's login flow
2. A list of ways to proceed is presented. In this case the choice is via mobile phone
3. The user is required to enter the username/password
4. A cronto image is generated and presented to the user. If the image is tapped, then the LuxTrust Mobile app should start
5. The user enters another password, specific to the LuxTrust Mobile app
6. Provided that the password is correct, an OTP is generated.
7. The generated OTP is sent back to the Retail app
8. The Retail app sends the OTP to Orely and waits for a response which contains the login status

Points 4 and 7 cannot be implemented with what UXP offers out-of-the-box and this is what this plugin was required for. It facilitates:

- Opening a third party app using a provided URI - this is supplied when tapping the cronto image
- Receiving a native message sent by a third party app and forwarding it to the front-end which posts it to Orely

3 Components

There are three parts of this feature:

1. Changes made to SPPIframe.tpl
2. SPServletFilter
3. UXP plugin

3.1 SPPIframe.tpl

This file contains javascript logic for interacting with the plugin. It is loaded in a Display Text in the Presentation editor. It needs to be added in the Phase where the login needs to be done.

In the Retail project it is loaded inside the LuxTrust Component. You can find the LuxTrust.ifp file in the folder LuxTrustRetail\Components\Common\Integration\LuxTrust. You can also navigate to it from the opened Solution: Retail -> Login ComponentPhase -> LuxTrustLogin ComponentPhase.

Once you navigate to the LuxTrust Component, in the LuxTrust Process -> Login Phase there is a Text Display Item. This loads a file called SPPiframe.tpl located in the solution's templates folder (LuxTrustRetail\Solutions\Retail\templates).

3.2 SPServletFilter

This comes in the form of a new library added to the Solution's lib folder: luxTrustSPServletFilter.jar. It can be found in LuxTrustRetail\Solutions\Retail\WEB-INF\lib. This contains logic that adds javascript to close the in app browser containing the WKWebView. This jar also contains the source code used to create it, in case a change is needed.

The javascript injected by the filter should only have an effect on the iOS build containing the plugin. Any other plaform should not be influenced by it.

To load this filter, a set of tags were added (see below) to the Solution's web.xml file found in LuxTrustRetail\Solutions\Retail\WEB-INF

This <filter> tag needs to be placed in a certain place within web.xml!

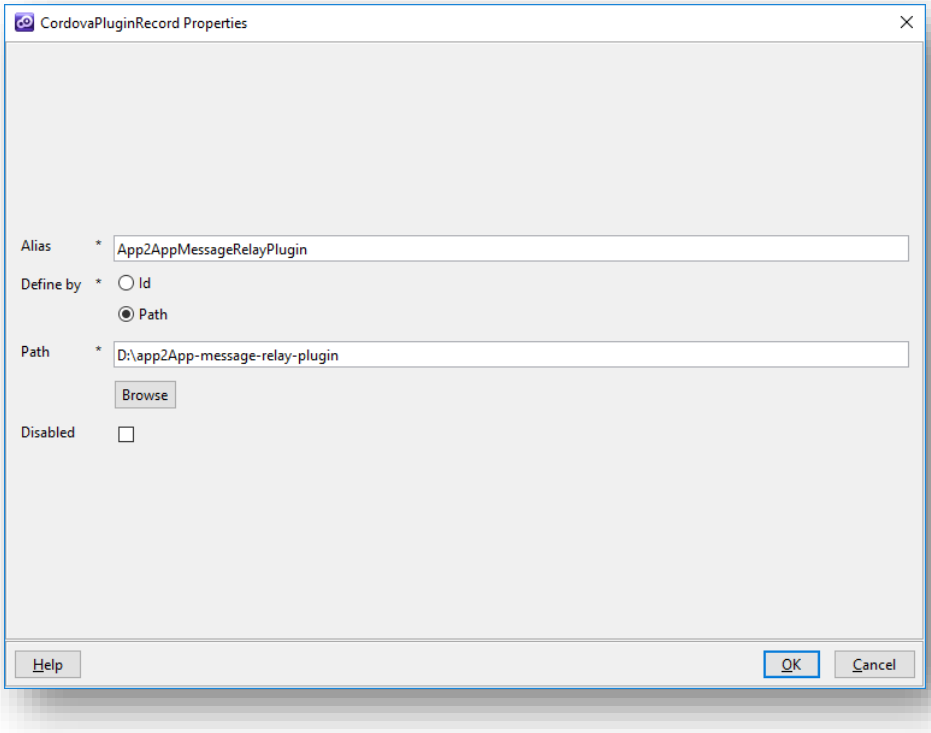
```
<filter>
	<filter-name>SPServletFilter</filter-name>
	<filter-class>com.edgeipk.servlet.SPServletFilter</filter-class>
</filter>
<filter-mapping>
	<filter-name>SPServletFilter</filter-name>
	<url-pattern>/saml/response/*</url-pattern>
</filter-mapping>
```

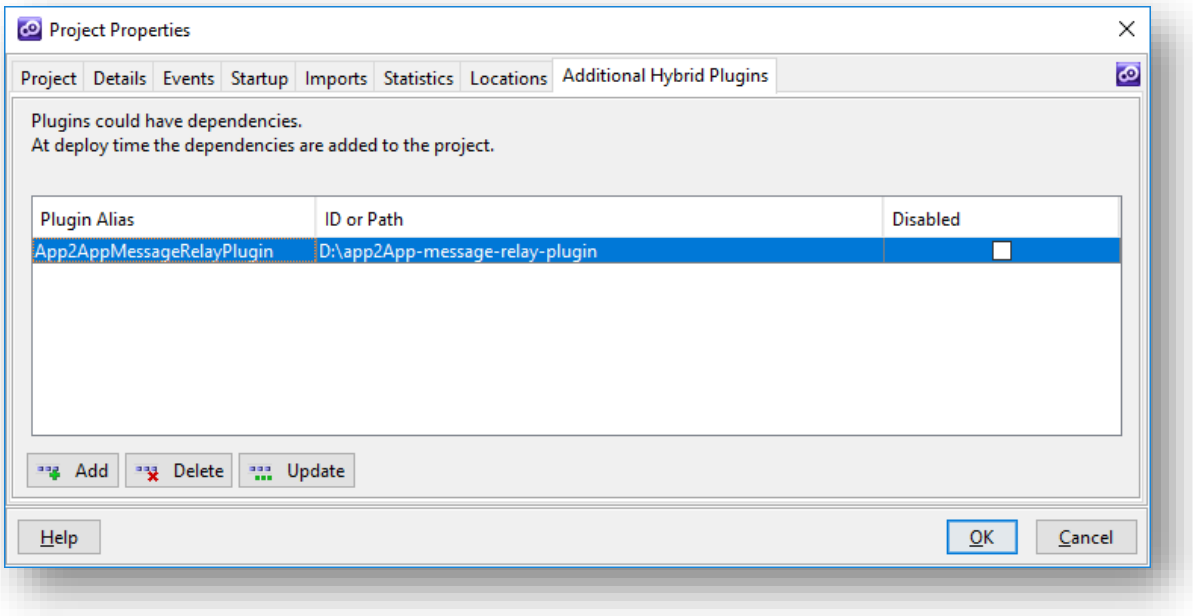
3.3 UXP Plugin

3.3.1 Installation

Add Cordova plugin via edgeConnect developer IDE. Installation will happen during the build process. Below are the steps.

1. Copy app2app-message-relay-plugin in your local computer (e.g. D drive)
2. In edgeConnect developer IDE, go to Project -> Properties… -> Additional Hybrid plugins
3. Click Add button. In Plugin Properties Window, give a desired alias and then choose Path radio button. Navigate to the app2app-message-relay-plugin folder on D drive you copied earlier. Please find the below screen shot for reference.





4 UXP installation

For the project to run properly, it needs to communicate using https. For this, some changes were done to the UXP installation.

1. The JVM was patched with the JCE patch. This means that files in the jre\lib\security were replaced with a set of libraries provided by Oracle which allow for stronger encryption.
2. edgeEmbeddedHttpsServer.jar was added to 5_4_1__18_LuxTrust/lib
3. edgeEmbeddedHttpsServer.jar was added to lax.class.path property in Developer.lax

   lax.class.path=../../lib/edgeEmbeddedHttpsServer.jar;../../extlib/commons-codec-1.4.jar; ...
4. Add ServletContainerClassForRun property in Developer.ifw. Note that Developer.ifw may be located in the Windows user's folder.

   ServletContainerClassForRun=com.acquire.intelligentforms.ide.presentationeditor.runpreview.JettyHttpsServletContainer

5 How to run the project

5.1 Run the project

To be able to run the project a couple settings need to be made in Retail.dsf:

KEY_STORE_PATH - the path to the keystore.jks file. It should be in the same folder as this README file.

KS_PASSWORD - the password for keystore.jks. The current one is "password"

KS_ALIAS - the alias of the key pair used to communicate with Orely (LuxTrust's server). The current one is "identity_new"

KS_ALIAS_PASSWORD - the password for the alias. Current one is "password"

LUXTRUST_RETURN_HOST - this should be your server's machine Host Name/url/ip. The Host Name can be found found in cmd, using the command: ipconfig /all.

5.2 Avoid SSL errors

The Retail project can only run using https. To avoid any SSL errors when browsing the site, a certificate needs to be created for the machine that runs the application and it needs to be added to the trust store on the local machine.

The following are the steps you need to follow:

You need to generate a private key for your machine:

Requirements:

1. OpenSSL (for win64: https://slproweb.com/products/Win32OpenSSL.html)
2. Keytool (this I believe comes built-in with Windows)
3. As always, make sure you have backups of everything

Steps:

1. Generate a private key for the new domain:

openssl req -new -x509 -extensions v3_ca -keyout myca.key -out myca.crt -days 1825

- Some questions will be asked; the only important part is Common Name: <this is where the Host Name from the previous step goes>
- For simplicity, I suggest using password=password for all we do during this demo
- This creates a private key (myca.key) and certificate (myca.crt)

2. Combine private key and certificate into a PKCS file:

openssl pkcs12 -export -out my.pfx -inkey myca.key -in myca.crt -certfile myca.crt

- This combines them into my.pfx

3. Import private key into Jetty keystore:

keytool -importkeystore -deststorepass password -destkeypass password -destkeystore keystore_https.jks -srckeystore my.pfx -srcstoretype PKCS12 -srcstorepass password -alias 1

- Keytool cannot import only a private key, that is why we had to combine them in a PKCS file
- This will import the private key into keystore_https.jks; this should be the keystore used by Jetty, inside the whatsBeenDone\http folder

You can add myca.crt as a trusted certificate in your browser/phone/etc, so it will never throw any SSL warnings. Note that SSL warnings are also network-dependent.

5.3 Certificates

For now, all the certificates and keys are up to date. The following is a list of certificates used by this project:

- LuxTrustRetail\Solutions\Retail\WEB-INF\lib\LTCertificateValidator-1.0.1.jar\TrustAnchors. The certificates included here are used to allow the Retail server to trust Orely.

- keystore.jks contains LuxTrust certificates and the "identity_new" key. Its password is "password"; It is used by Retail to send data to the Orely server. It is specific to the Orely server that provides the authentication service. This means that a different certificate needs to be used in production.

- http\keystore_https.jks - this contains the myKey that was created previously. It is necessary for the server that runs the Temenos UXP project. It allows the creation of secure connection with the server's clients.

5.4 Configuring plugin.xml

We need certain configuration before we install this plugin with hybrid solution. These Configurations are part of plugin.xml file. Below are the details.

1. **CLIENT_URL_SCHEME** – This is issued by LuxTrust for the service provider. It is mandatory and used during the communication between the two apps.
2. **IFRAME_NAME** – when the LTMOB responds to SPMOB with the OTP, a message containing this OTP will post from an iframe with this name. Please note that we can implement the integration without an iframe too. In this case, the value must be "NONE" so that the app will understand the integration does not use an iframe.
3. **ORELY_URL** – this is the URL to access the Orely server. For test and production environment, they use different urls.

Below is the screen print from plugin.xml, which shows the details

```
<!-- Please provide the url scheme for service provider app, default value is 'SP0022' -->
<preference name="CLIENT_URL_SCHEME" default="SP0022" />
<!-- Please provide the name of IFRAME used in the app for LuxTrust integration, default value is 'orelyIframe'.
     Input "NONE" if no iframe integration is required. -->
<preference name="IFRAME_NAME" default="orelyIframe" />
<!-- Please provide the Orely url, for Testing: 'https://orely.test.luxtrust.com' & for Production: 'https://orely.luxtrust.lu'  -->
<preference name="ORELY_URL" default="https://orely.test.luxtrust.com" />
```

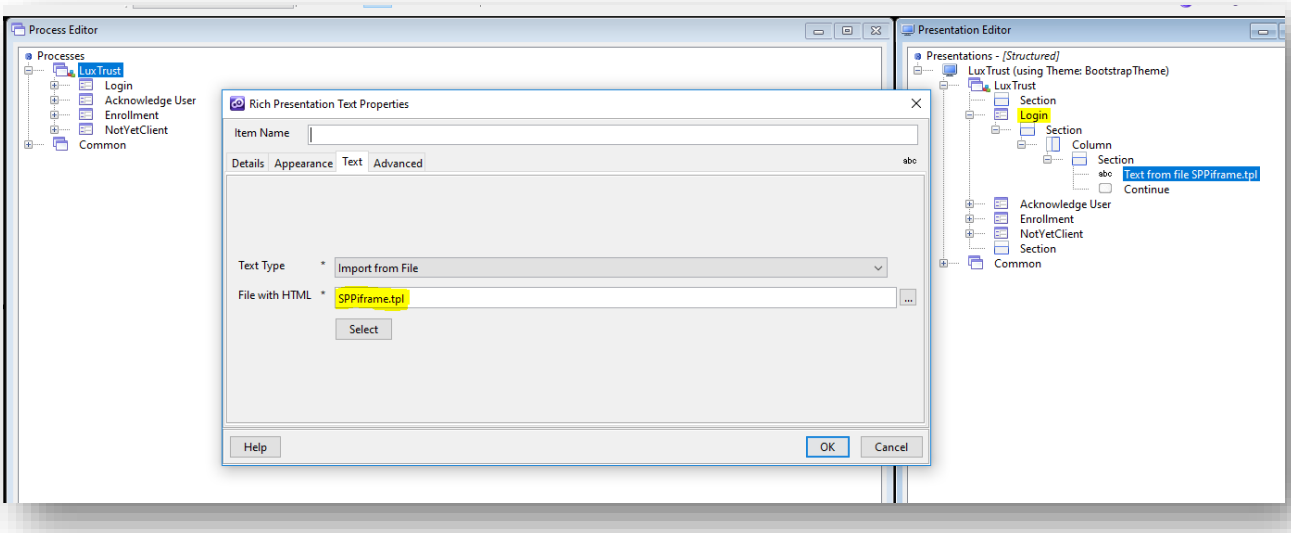
Please note that these configurations are common for both Android and iOS.

6 Patching the existing solution

To patch an existing solution, the components mentioned in chapter 3 need to be installed.

6.1 SPPIframe.tpl

This needs to be loaded in a DisplayText. The DisplayText item has to be placed in the Phase that displays the login page. See below an example of how this might look in a project.



This template behaves differently depending on the platform on which the page is displayed. If the current platform is not iOS, then startLuxtTrustAuth.html page is loaded within an iFrame and this triggers the loading of the LuxTrust-provided content. If the current platform is iOS, then an in app browser using a WKWebView is opened and this loads LuxTrust-provided content. Note that this WebView will only contain information provided by the LuxTrust server and the page has no UXP-related knowledge. We also added another html page if the customer need some sort of branding. “startLuxtTrustAuthBranding.html” internally use an iframe to load startLuxtTrustAuth.html and it has the capability of adding additional html elements like images etc. for branding purpose. This is exactly similar to integrate orely portal with an iframe.

6.2 luxTrustSPServletFilter.jar

It is just a Servlet filter that adds some javascript that closes the in-app browser which contains the WKWebView. Since the separate in-app browser is only opened in iOS, this filter should not affect any other platforms.

The javascript injection is only done to the configured `<url-pattern>` responses. It must target the URL for the response of the authentication request made to Orely. To add it to an existing UXP project, two steps need to be done:

1. The following tag needs to be added to the web.xml file (located in the WEB-INF folder of the Solution):

```
<filter>
	<filter-name>SPServletFilter</filter-name>
	<filter-class>com.edgeipk.servlet.SPServletFilter</filter-class>
</filter>
<filter-mapping>
	<filter-name>SPServletFilter</filter-name>
	<url-pattern>/saml/response/*</url-pattern>
</filter-mapping>
```

Note that the `<filter>` tag needs to be added before any `<listener>` tags.

2. The luxTrustSPServletFilter.jar file needs to be added to the Solution’s lib folder (located in the WEB-INF folder)

6.3 App2app message relay plugin

This plugin needs to be included in the Solution project. Its inclusion will only have an impact when a hybrid app is built because the build process will also include this plugin and is not used when simply deploying the Solution. The more detailed installation instructions can be found in chapter 3.3.1 of this document.

6.4 Mobile app

When this plugin is added to an app, a rebuild of the app must be made and distributed to the users. This means that they need to update their app using their respective app store.

7 Platform specifics

7.1 iOS

The main problem that needed to be overcome for iOS was the fact that the login flow could not be displayed in an UIWebView. This is what Cordova uses and UXP uses Cordova for its hybrid apps. To overcome this obstacle, the plugin opens an in-app browser which uses a WKWebView. This makes this problem specific for iOS as UIWebView/WKWebView are iOS-specific implementations and are not present in Android.

7.1.1 How to use

We can open WKWebView from java script by using below method.

cordova.InAppBrowserWKWebView.open(authUrl, '_blank', 'location=no')
authUrl – Is the url to open in WKWebView
location=no – Decides whether to show location/parh in the user interface.

7.1.2 Notes

When the WKWebView is opened, it is not aware of the UXP underlying session.

The SPServletFilter only affects the flow of the client devices using iOS. It is only necessary for the iOS app to work properly.

Below screen-shot shows the iOS configurations.

```
<platform name="ios">
    <js-module src="www/inappbrowserwkwebview.js" name="inappbrowserwkwebview">
        <clobbers target="cordova.InAppBrowserWKWebView.open" />
        <clobbers target="CLIENT_URL_SCHEMEwindow.open" />
    </js-module>
    <config-file target="config.xml" parent="/*">
        <feature name="InAppBrowserWKWebView">
            <param name="ios-package" value="CDVInAppBrowserWKWebView" />
        </feature>
    </config-file>
    <config-file target="*-Info.plist" parent="CFBundleURLTypes">
        <array>
            <dict>
                <key>CFBundleURLSchemes</key>
                <array>
                    <string>$CLIENT_URL_SCHEME</string>
                </array>
            </dict>
        </array>
    </config-file>

    <config-file target="*hybridConfig.plist" parent="app2appConfiguration">
        <dict>
            <key>clientUrlScheme</key>
              <string>$CLIENT_URL_SCHEME</string>
            <key>iframeName</key>
              <string>$IFRAME_NAME</string>
            <key>orelyUrl</key>
              <string>$ORELY_URL</string>
        </dict>
    </config-file>

    <header-file src="src/ios/CDVInAppBrowserWKWebViewUIDelegate.h" />
    <source-file src="src/ios/CDVInAppBrowserWKWebViewUIDelegate.m" />
    <header-file src="src/ios/CDVInAppBrowserWKWebView.h" />
    <source-file src="src/ios/CDVInAppBrowserWKWebView.m" />

    <framework src="CoreGraphics.framework" />
</platform>
```


TEMENOS
The Banking Software Company

7.2 Android

When using Android, the page behaves similar to the desktop version in regards to loading content from Orely. An iFrame is used and it communicates with the main page via the “postMessage()” mechanism. All this plugin does is to provide a way of opening an app using an URL and to return the result from that app back to the javascript that provided that URL

The SPServletFilter should have no effect on the workflow of the application running on Android devices.

Below screen-shot shows the Android configurations.

```
<platform name="android">
    <config-file target="AndroidManifest.xml" parent="/manifest/application/activity/intent-filter/action[@android:name='android.intent.action.MAIN']/../..">
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT"/>
            <category android:name="android.intent.category.BROWSABLE" />
            <data android:scheme="SP0024" />
        </intent-filter>
    </config-file>
    <config-file target="config.xml" parent="/widget">
        <feature name="App2AppMessageRelayPlugin">
            <preference name="IFRAME_NAME" value="orelyIframe" />
            <preference name="ORELY_URL" value="https://orely.test.luxtrust.com" />
            <param name="android-package" value="com.temenos.connect.app2app.App2AppHooks" />
            <param name="onload" value="true" />
        </feature>
    </config-file>

    <source-file src="src/android/com/temenos/connect/app2app/App2AppHooks.java" target-dir="src/com/temenos/connect/app2app" />
</platform>
```


TEMENOS
The Banking Software Company