Silver Mobile
User Guide 2.1

Silver Mobile

Silver Mobile is a platform for running HTML5-based mobile web apps within a managed device and server runtime environment. Enterprise capabilities are provided via built in services and APIs. Security, portal, real-time data, and native device services are accessible via JavaScript and Java APIs for the client and server.

TIBCO Software Inc 3307 Hillview Ave. Palo Alto, CA. 93404

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

TIBCO, The Power of Now, TIBCO Silver, TIBCO Silver Fabric, TIBCO Silver Mobile, TIBCO ActiveMatrix Service Grid, TIBCO Rendezvous, TIBCO Administrator, TIBCO Enterprise Message Service, TIBCO InConcert, TIBCO Policy Manager, TIBCO Runtime Agent, and TIBCO Hawk are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

EJB, Java EE, J2EE, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

TIBCO products may include some or all of the following:

Software developed by the Apache Software Foundation (http://www.apache.org/). Software developed by Joe Walnes and Xstream Committers.

Software licensed under the Eclipse Public License. The source code for such software licensed under the Eclipse Public License is available upon request to TIBCO and additionally may be obtained from http://eclipse.org/.

Software developed by the OpenSymphony Group (http://www.opensymphony.com/).

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE

Table of Contents 4

1 Product Overview

1.1 Introduction

An existing HTML5-based mobile web app deployed into Silver Mobile gains immediate enterprise capabilities. Authentication, authorization, user access control, and a native look are immediately available to the app without modification. A JavaScript API can be used to access asynchronous, real-time data services; native device capabilities such as camera, contact, and notification services; as well as security and customization services.

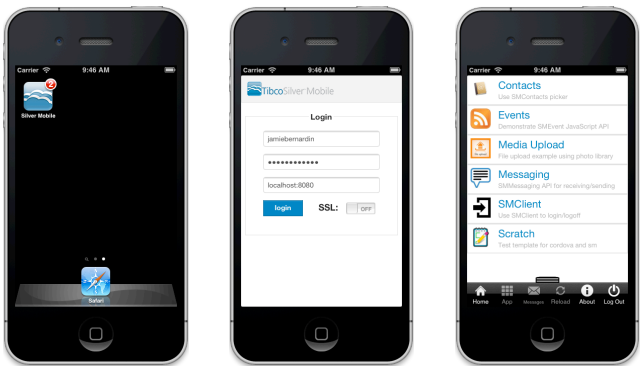
1.2 Runtime Architecture

Silver Mobile consists of a native client container (available at the App Store or Android Market) and server consisting of Java services running within a Jetty web server.

Client

The Silver Mobile Client is an actual native mobile app. When it is opened, the user is presented with either the default Silver Mobile login screen or a fully customized start page/app. In the default mode, after the user logs in successfully, a view of the accessible enterprise app(s) is displayed.

The simple login screen can be customized considerably. For example, the client can allow an anonymous user access to a working UI without a valid login. Different apps or sections can then be optionally protected. Please see the section on customization for further details.



Server

The Silver Mobile Server(cluster) is a Java process that communicates with the mobile clients over HTTP(S) and provides an administrative web interface for managing the system. This interface is used to provide a number of administrative functions like managing users, apps and security as well as testing and monitoring messages and events. These features will be covered in the next section.

2 Key Concepts and Capabilities

2.1 Security

Users and the Security Realm

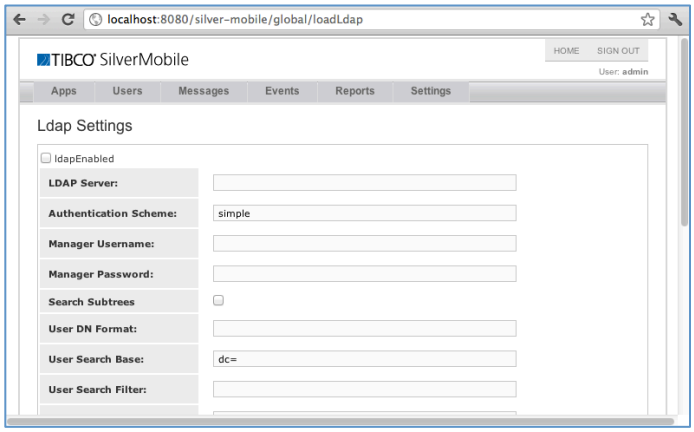
A User is a mobile app client or administrative console user. The administrative database (configured in SilverMobile/conf/silver-mobile.properties) has a User table that holds both role and application access permissions for all users. This table can be populated in four ways:

- An LDAP (Active Directory) setting is configured in the system. When a user attempts to log in, the credentials are forwarded to the LDAP system. If successful, the username is used to find the corresponding entry in the user table. If not found, a new entry is automatically created.
- A user can be inserted into the RDBMS using any script, batch, or ad hoc UI tool.
- The Silver Mobile web admin can be used to create new users.
- Any external authorization hook can be created in Java that inserts new users into the database automatically.

Note that if an external authorization provider is used, the internal database will not actually hold credential information. The database only stores additional Silver Mobile related information.

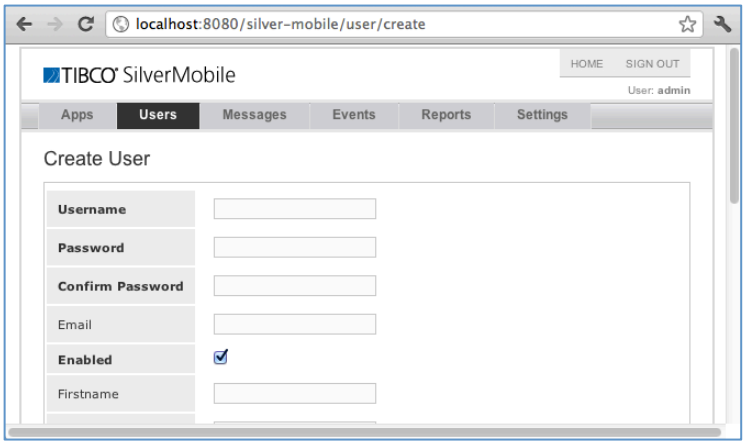
Authentication

For integration with existing enterprise LDAP systems, the configuration can be enabled using the administrative web console as shown below.



For integration with other systems, an External Authentication Hook can be created using the Silver Mobile server-side Java. This approach allows Silver Mobile to delegate authentication to another security/identity system after it's properly configured in the 'settings' section of the admin tool.

The administrative database can also hold password-based credentials. These passwords are stored (and compared) with a properly salted SHA256 digest. Users can be created manually using the admin page shown below:



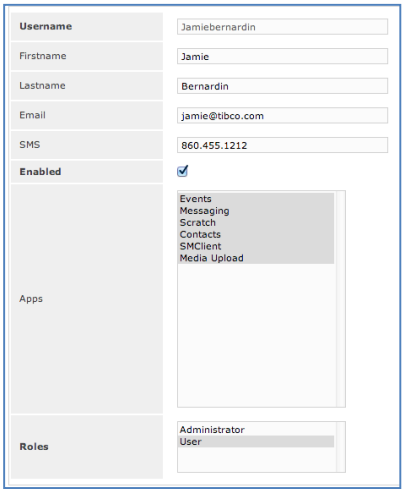
Case Insensitive User Name

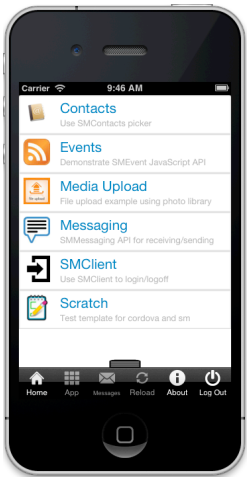
If the LDAP uses case insensitive authentication, the "Case Insensitive Username" option should be enabled for the Silver Mobile Server. "Case Insensitive User Name" can be enabled in the Settings/System page on the Admin Console. All user names will be created in lower case in the internal Database when "Case Insensitive Username" is enabled.

Authorization

A user can only view and use mobile apps that have been assigned to the user. If the user does not have privilege to use an app, it will not be visible from the Silver Mobile Client portal view. Silver Mobile provides a security filter that can be added to the mobile web application to protect resources from unauthenticated and unauthorized users. The authorization is accomplished in two ways: 1. Via direct modification of the user table. 2.

Via the LDAP or external group mapping in the app data (which will be covered in a later section). The first direct method is shown below:





App Security Context

Silver Mobile can provide a security context to deployed apps if a security filter is added to each web.xml file as shown below:

```
<filter>
  <filter-name>ShiroFilter</filter-name>
  <filter-class>
    org.apache.shiro.web.servlet.IniShiroFilter
  </filter-class>
  <init-param>
  <param-name>configPath</param-name>
  <param-value>file:conf/shiro.ini</param-value>
  </init-param>
  </filter>
<filter-mapping>
  <filter-name>ShiroFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

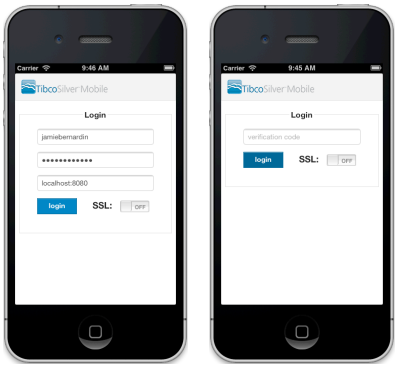
In addition to providing authentication and authorization of the web app itself, the filter will also create a valid security principal for use within any server-side Java API allowing for programmatic authorization in web services.

Two-Factor Authentication

Users can be required to provide an additional passcode when logging into the client. This can be accomplished within the configured LDAP security

realm by implementing the Java ILDAPTwoFactorExtensionHook. This is where the implementer can decide to use SMS or email for passcode delivery. If security is being provided by the IExternalAuthenticatorHook, then this class can be expanded to handle two factor login. The first attempt at login should return a Map with the field REQUIRES_2_FACTOR variable set to true. Passcode generation and delivery is a responsibility of the implementer of this class.

The default client login will automatically change UI inputs to accommodate the second passcode to be entered. For customized clients, please refer to the SMClient JavaScript API section for login and lifecycle control. The default UI sequence when two-factor authentication is in effect is shown below.



2.2 Apps

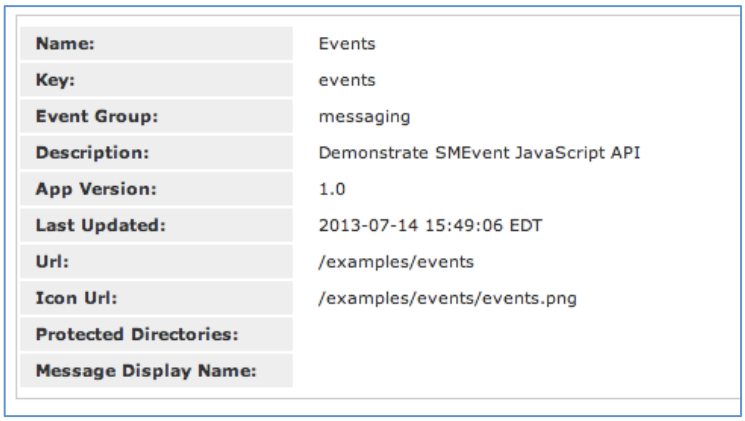
Creating and Deploying

Apps are created using HTML5/JavaScript directly or by using the various JavaScript toolkits such as SenchaTouch, JQueryMobile, etc. The optional server-side components (JSON services) can be created using any Java servlet approach (Jersey, Groovy, etc). Like any web-based application, these artifacts should be bundled into one folder and deployed to the SilverMobile/webapp directory.

Provisioning

An app entity in SilverMobile is typically created by using the web admin tool. The required properties are the name, key, icon and URL of the mobile web app. The other meta-data and authorization properties are

covered in related sections. The URLs for the app and icon can be relative to the 'SilverMobile/webapps' directory or an absolute (external) web address.



Name:	Events
Key:	events
Event Group:	messaging
Description:	Demonstrate SMEvent JavaScript API
App Version:	1.0
Last Updated:	2013-07-14 15:49:06 EDT
Url:	/examples/events
Icon Url:	/examples/events/events.png
Protected Directories:	
Message Display Name:	

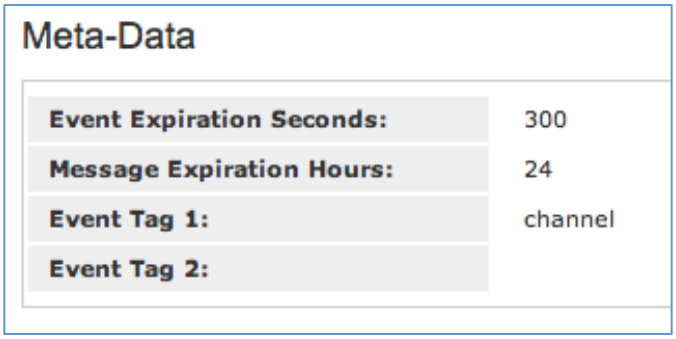
2.3 User Messages

Each application and user combination has its own message queue; which means that each user message has a destination target given by the appkey and username combination. Messages can be both human readable and/or carry rich JSON objects for processing within the app. If the default native UI message box is being used (see picture below), the JSON message must contain a 'subject' property. For custom UIs, any JSON object can be used with or without populating the subject and body properties. Please see the Java and JavaScript APIs for more details. With the default Silver Mobile client, the native message list and viewer are automatically available for each deployed mobile web app. A message can be read and deleted using this interface. Alternatively, these native UI components can be disabled and the app can provide its own message viewing interface using the JavaScript Message API.

Message Message

2.4 App Events

An app event is a unit of real-time data that could be relevant to a broad class of app users. Examples include stock updates, traffic alerts, promotional sales, etc. The raw data, contained in the 'body' element of the event, can be consumed in charts, tables, alerts, and so forth. The lifecycle of this data is determined by the expiration time indicated in the meta-data section of the app provisioning screens:



The Event Tag fields determine which JSON fields will be used for indexing by the SilverMobile Server. This way, the asynchronous JavaScript event listeners can specify criteria (basically a "where" clause in SQL dialect) for which they will elect to receive update events. The above screen shot is the meta-data associated with the event example app. In this case, the app user can select which news events to listen for. By specifying this property in the server admin, Silver Mobile is able to index the 'channel' property of the posted events. This is covered in more detail in the JavaScript API and example sections.

Event Groups

By default, apps only listen for events that are posted with their appkeys. This makes it difficult for different apps to share events with other apps. To combine multiple apps into one event group, use the Event Group property. All apps that share the same Event Group value (any valid string) will automatically post and listen to those same events.

2.5 Silver Mobile Client

Styling

The color scheme for the native Silver Mobile UI message list and single message view components may be customized on a per-app basis. To do so create a new Style in the administration console and specify the desired color for each individual UI element. Colors are specified as CSS hex color values, i.e. #0F0F0F. Any unspecified color defaults to the native color for that element. The style may then be applied to any app in the app editing screen.

Offline Mode

The Silver Mobile Client will fully function offline. The user credentials are encrypted and stored on the device in case there is no connectivity to the server. If the deployed HTML5 apps have caching enabled (either through direct creation of a cache manifest or through a toolkit which provides equivalent functionality) and they have been used once online, then they will be available offline. When the client is in offline mode, it will periodically try to re-establish a live connection in the background without interruption to the user.

While offline the user will not receive any messages or events from the server. The user will also not have access to any apps that were not previously cached by the client, and will not be notified when s/he is granted access to new apps.

Single-App Mode

When a user only has access privilege to use only one app, the Silver Mobile Client will proceed from the login screen directly to the app, without stopping at the portal screen (app list page). Note that the portal screen remains unavailable even if the user is granted access to additional apps while connected. The user must log off and then log back on in order to gain access to the portal screen and newly added apps.

Customizing the Silver Mobile Client with a remote startPage can create a more advance version of Single-App Mode. When the client is started, it will immediately render the remote app and the user will not yet be logged in (anonymous). Login can be activated by the app when it is deemed appropriate. When running in the browser and using smengine (the native runtime equivalent for the browser), you can set a configuration variable to make the environment run in single-app mode explicitly. See the section on smEngineConfig for more detail.

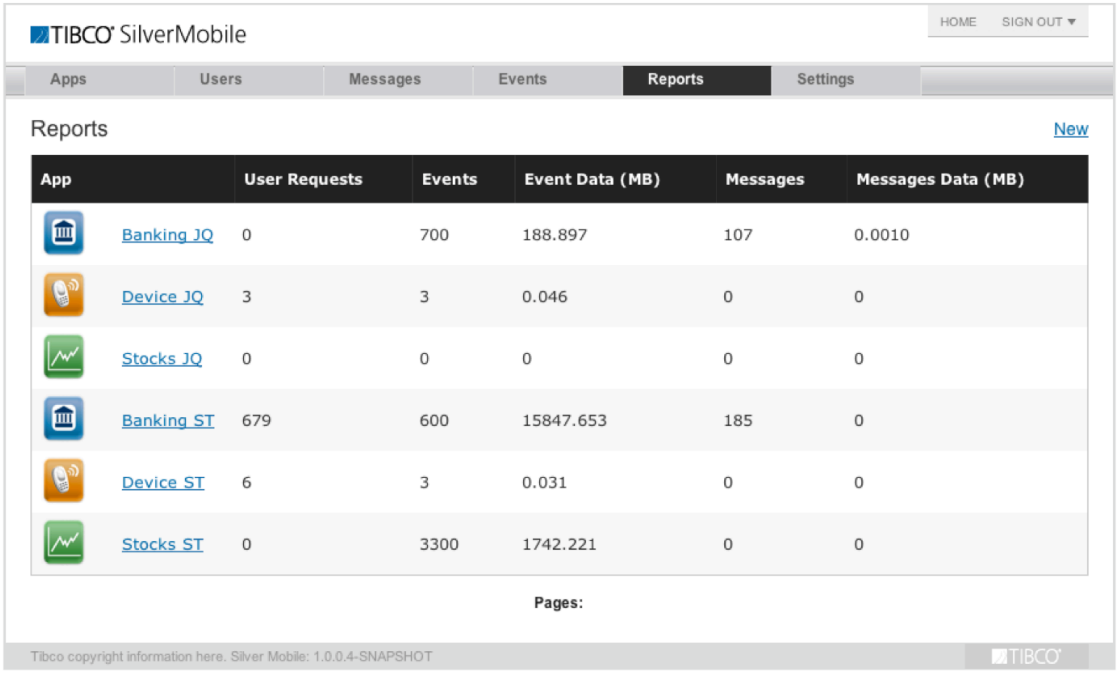
Fixed Server Mode

If a fixed server URL is specified in the client's customization properties (see the Client Customization section) then the login screen will not need to present the server address or SSL switch fields. The client always connects to the same URL.

2.6 Admin Utilities

Usage Reports

It is possible to get a global view of all the user requests, events, and message data by going to the “Reports” section of the admin tool.

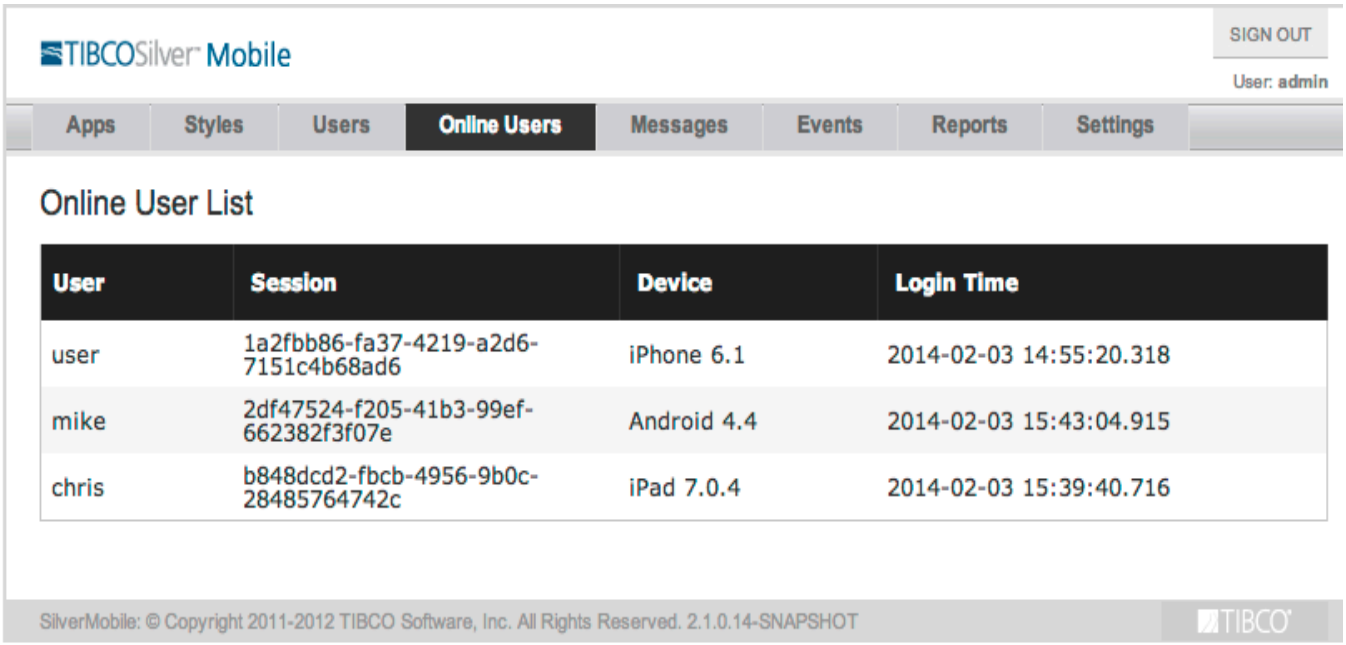


Audit Table

A table is provided in the Silver Mobile Admin database that logs all changes made to the system:

Online Users

All users currently logged in with an active session are displayed in the admin console. Users will be displayed by session since it is possible for a single user to be logged into many mobile or browser sessions. Admin console users are not displayed.



3 Client Javascript API

3.1 Overview

Mobile web apps running in the Silver Mobile Client have access to a variety of extended device, messaging, and lifecycle functionality via the Silver Mobile JavaScript API as well as the Cordova (Phone Gap) runtime. All mobile web apps should include the file sm-cordova.min.js, e.g.:

```
<script type="text/javascript" src="../../js/sm-cordova.min.js"></script>
```

This file provides JavaScript APIs to the native functionality contained in both Cordova and Silver Mobile.

The API calls do not support synchronous return of results (with the exception of SMClient.isLoggedIn()). The uniform API call signature usually takes one JavaScript input object containing one or more properties, plus separate success and error callback functions as arguments. If no error callback is specified, the callback registered in the SMClient is used – see the error handling section below.

> ***Notes***: Dates are always in GMT and expressed as strings of the following format: "yyyy-MM-dd'T'HH:mm:ss'Z'" i.e. 2011-06-01T17:15:30Z".
>
> The iOS JavaScript bridge is in some cases more stringent than the Android JavaScript bridge. Certain improperly formed JavaScript calls will work on Android *but not on iOS*. For example, JavaScript sub-objects passed in as strings will be interpreted as JSON objects by the Android bridge, but will be treated as strings by the iOS bridge.

3.2 Initialization

When a web app is loaded the Silver Mobile Client initializes the JavaScript API and binds the API objects to native client functionality. As this initialization is not instantaneous, the JavaScript API is not available to the web app immediately. Once the framework has been initialized, the Client will fire a 'ondeviceready' JavaScript event. If the app needs to run JavaScript code that accesses the Silver Mobile API when the page is loaded it must be executed in response to the 'deviceready' event, rather than in response to 'onload'. Typically this is where the app will register event and message listeners.

Example:


```
// Wait for Cordova and Silver Mobile  to load
//
document.addEventListener("deviceready", onDeviceReady, false);

function onDeviceReady() {
    SMMessaging.addMessageListener(onMessage);
}
```

3.3 Error Handling

Web apps can register a global JavaScript API error handler via SMClient.registerErrorHandler(errorHandler). If the client encounters an error while handling a JavaScript API call (for example, if a function call is missing required arguments) the client will invoke this error handler as a default if no callback is passed as input to the actual function. The error handler callback receives an error object as its sole argument.

Additional error codes may be added at a later date, so for forward compatibility error handlers should take into account that an unknown error code may be sent from the system.

Error Object

- *code* - The general error code (see below)
- *message* - The error message
- *interfaceName* - The interface of the failing method call
- *method* - The name of the failing method
- *originalArguments* - The original arguments that were passed into the method as a JSON object. These arguments may not be available if malformed arguments were passed to the method: i.e. passing a string instead of a JSON object, or passing the wrong number of arguments.

Error Codes

- *1* – Missing or bad argument(s)
- *2* – Internal error
- *3* – Offline – functionality not available
- *4* – Operation canceled by the user
- *5* – Connection error
- *6* – Current application removed

3.4 Phone Gap

The Silver Mobile 2 client includes the Cordova runtime engine and provides dynamic access to the full set of device APIs. The example applications provide a skeleton app called “scratch” that can be used to try all the cordova examples found here:

http://cordova.apache.org/docs/en/3.1.0/

Simply cut and past any of the full examples into the index.html file in the scratch directory and all native calls will work within the Silver Mobile client.

4 Server-side Java APIs

4.1 Overview

The Silver Mobile server-side API is primarily used for sending user messages or posting application events. There are only a few service calls and they are straightforward. One can post raw JSON data to HTTP service locations, but managing SSL, session, and cookies is not trivial for daemon processes that might run intermittently. For this reason, a simple Java API is provided that takes objects, maps and vectors for events, messages, and batch events and provide the JSON conversion, session management, and request-response functions. The API and a few examples are included in the SDK folder, so we only highlight a few points here. The Groovy runtime jar is required to use the server-side API and the readme.txt file included in the SDK/example-scripts directory explain the requirements for compiling and running the examples.

4.2 Establishing a Connection

After creating and initializing a Connection object, it will maintain a valid session or re-login to establish a new session automatically. The example invocation can be used in a batch procedure or as a long-lived object within a service or daemon process:

```
SMConnector smConnector = new SMConnector("http://localhost:8080", "admin", "admin");
```

One can also use SMConnector from within a servlet deployed to the Silver Mobile Server. In that case, you will not need to provide any credentials, since ServletRequest has a valid principle - assuming it's contained within an app and the client has logged into the Silver Mobile Client. See the 'mediaUpload' example and the Java API docs for more information.

4.3 Sending User Messages

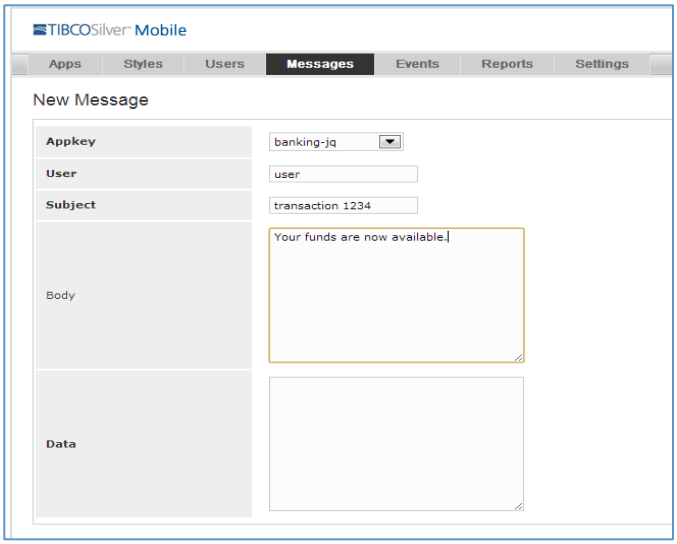
A user message is sent to a particular app and user combination, so these properties are required in the input object. In addition to these two fields, the message can contain a subject and body property – each of which must be String types.

```
Map<String, Object> message = new HashMap<String, Object>();
message.put("appkey", "messaging");
message.put("username", "user");
message.put("subject", "hi from java code!");
message.put("body", "This will show up in the native UI.");
smConnector.sendMessage(message);
```

By including subject and body properties, the message will be automatically rendered in the default native message views. You can disable the mailbox or avoid using those property names if you do not wish to have messages automatically rendered. You can send additional properties either way by using the 'data' property, as in the example below:

```
Map<String, Object> message = new HashMap<String, Object>();
message.put("appkey", "messaging");
message.put("username", "user");
Map<String, Object> data = new HashMap<String, Object>();
data.put("foo", "bar");
data.put("random", Math.random());
message.put("data", data);
smConnector.sendMessage(message);
```

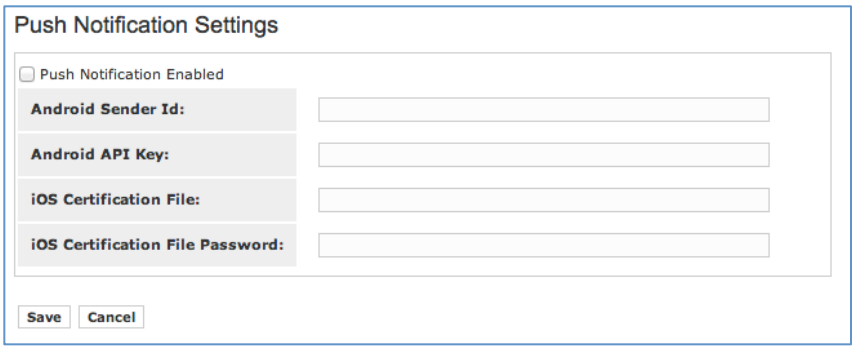
For testing purposes, the web admin tool allows administrators to send and view messages. The screen shot below demonstrates a simple test message being sent to a specified user and app.



4.4 Unified Push Notification

Messages sent using SMConnector class in Java or SMMessaging namespace in JavaScript have the ability to forward part of the message to the Android cloud and Apple push notification services. The message testing page in the web admin demonstrates the extra parameters required.

In order to configure the server you will need to visit both the Google Cloud Messaging center and Apple's developer portal to obtain the appropriate credentials. Configure the 'Push Notification Settings' with these credentials in the web admin tool as shown below.



Note for iOS, the actual rebranded Silver Mobile Client will need to be recompiled with the private key obtained through the Apple site while Android does not require that step.

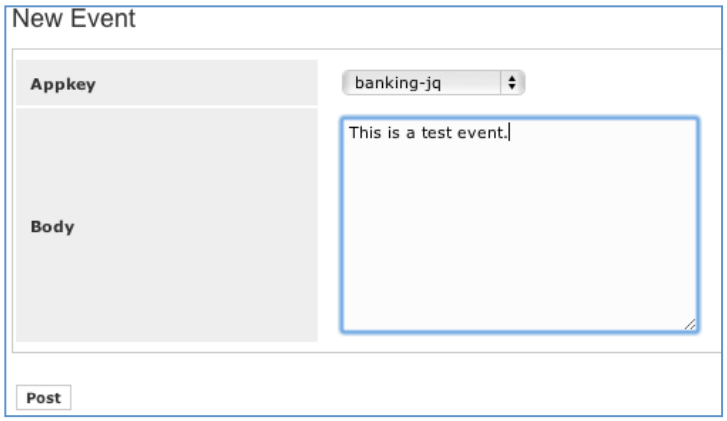
4.5 Posting App Events

App events are posted using input objects containing both an 'appkey' and 'body' property (the 'body' property is like the 'data' property in messages). For convenience and testing from the admin tool, the body can be a string and Silver Mobile will convert it to a Map with property named "value". This way, the event listener will always receive a Map object as the body. If the target app belongs to an event group, the event will be sent to all apps in the group.

The example below illustrates a complex event being posted to Silver Mobile via the Server-side API.

```
Map<String, Object> event = new HashMap<String, Object>();
Map<String, Object> body = new HashMap<String, Object>();
body.put("news", "Event posted from Java");
body.put("channel", "national");
event.put("appkey", "events");
event.put("body", body);
smConnector.postEvent(event);
```

For testing purposes, the web admin tool allows administrators to post and view events. The screen shot below demonstrates a simple event posted to an app (and subsequent clients using that app).



4.6 Posting Event Batches

It's much more efficient to post a collection of Events. Doing so will help with submission overhead as well as the network layer because the network layer is involved in determining whether a suspended HTTP response should be resumed to fetch real-time data. This can be done just once for all events contained within the batch by setting the 'bodies' property to a list of maps. See Java and JavaScript documentation for more information.

4.7 Event and Message Listeners

The SMConnector Java API allows application services to receive asynchronous messages and events. This capability is particularly useful for listening to geo-location change events. A process can listen for location events and test proximity against store locations. Promotional messages can be sent via the data push APIs. Refer to the Java API docs as well as the example-scripts in the SDK folder.

5 Running Silver Mobile in the Browser

5.1 Overview

It is possible to use Silver Mobile inside a browser environment, rather than the native Silver Mobile Client (including mobile browsers). This is obviously critical for developing and debugging apps, but this capability can also be used to leverage Silver Mobile within the browser of mobile devices that we currently don't support fully (like Windows 8).

None of the native capabilities will work in a browser environment, but the events, messaging, security, lifecycle, and app management capabilities will work. The Silver Mobile JavaScript and Cordova objects are contained in the smengine.js file (debug and minified versions). You should always include this file in the index.html file (or equivalent) so that these namespaces are satisfied in the browser environment. If in fact, the app is running in the native Silver Mobile Client, the smengine objects will be ignored and the native objects will be injected into the JavaScript runtime.

5.2 Including Silver Mobile Engine in your app

The smengine files are located in the webapps/js directory of the Silver Mobile server. Include it in your app like any other JavaScript library, for example:

```
<script type="text/javascript" src="../../js/sm-cordova.min.js"></script>
<script type="text/javascript" src="../../js/smengine.debug.js"></script>
```

It is recommended that all of your mobile web apps contain both runtime engines – the first is for operation within our native clients and the second is for operation within a mobile browser. The relevant runtime engine will be used while the other will be ignored – so it's safe to include both files.

5.3 smEngineConfig

The Silver Mobile browser engine (smengine) can be tailored with the smEngineConfig variable. A default instance is defined in the beginning of smengine:

```
if (smEngineConfig == undefined) {
    // if undefined in app set to default Portal-Mode
    var smEngineConfig = {
        debug : true,
        singleAppMode : false,
        startPage : '/customization/login.html',
        portalURL : '/customization/index.html'
    }
};
```

This configuration instructs smengine to switch to the startPage if the user is not yet authenticated. Also, the portalURL page will be the screen shown after properly logging in. The pages located in the 'customization' area are intented to be modified if so desired.

If an instance of Silver Mobile Client is intended to be used for a single app only then this configuration can be modified in the app itself (imagine a retail use case for consumers and a separate portal use case for employees). The following snippet is from the example 'smclient' and demonstrates how to properly override smEngineConfig:

```
<script type="text/javascript">
    var smEngineConfig = {
        debug : true,
        singleAppMode : true
    };
</script>
<script type="text/javascript" src="../../js/smengine.debug.js"></script>
```

5.4 Providing implementation for debug and unsupported handlers

All native calls made with SM* objects (SMCamera, SMMedia, etc) and Cordova objects will not typically work in a pure browser environment. What action results when a user clicks a camera button (or equivalent) is up to the developer. For example, the default implementation causes a console log message to be written with the input functions to the unsupported methods. These can be overwritten in the smengine JavaScript files:


```
function _SMNotSupported (module, method, args) {
        //alert("The " + module + " is not supported in this environment.")
        if (smEngineConfig.debug) {
                _SMDebug(module + "." + method  + "(");
                var argStr = JSON.stringify(args)
                _SMDebug(argStr.substring(1,argStr.length-1))
                _SMDebug(")");
        }
}

function _SMDebug(msg)
{
        if (window.console != undefined && smEngineConfig.debug)
        {
                window.console.log(msg);
        }
}
```

6 Silver Mobile Client Customization and Rebranding

The Silver Mobile Client can be easily by modified by editing the HTML login page to use different colors, logo, style, etc. In this approach you can also customize some properties like fixing the start server URL, only use SSL, remove toolbar elements, etc.

For running in Single-App mode, the first screen that appears upon start-up can point to an actual application hosted on a server. So the start or landing screen is an actual mobile app. The user is not logged in, but can still use the app as an anonymous user. Many consumer-based mobile apps work this way – the user can check for retail locations, company hours, etc. In Silver Mobile, the login is controlled completely through a JavaScript API and is usually deferred until the user needs to perform a privileged action – like making a purchase or viewing account information.

The pros and cons for each approach depends heavily on the use case. For a consumer facing application the user probably only has privilege to use a single app, and there may also be good reason to defer login. In this case, the second approach is more flexible.

For employee-based mobile apps – where the portal functionality is mostly used – the first approach is preferred. The user is immediately presented with a login screen, and after logging in, they can choose which mobile app they would like to use out of those which they have access to.

6.1 iOS Customization

The Silver Mobile Client core comes with an iOS project creation script that can be used to start a project with the correct name, location, and bundle identifier. First expand the iOS client core 'SMC-iOS-SilverMobileCore.tar.gz' and cd into the bin directory. There is a creation script called 'create_project.sh'. Executing the command without arguments will show a brief description of the required and optional arguments.

Usage:

```
./create_project.sh <YourProjectName> <com.your.bundle.prefix>
<PathToCreatedProject> [<AbsoluteEmbeddedFrameworkPath>
<YourCompanyName>]
```

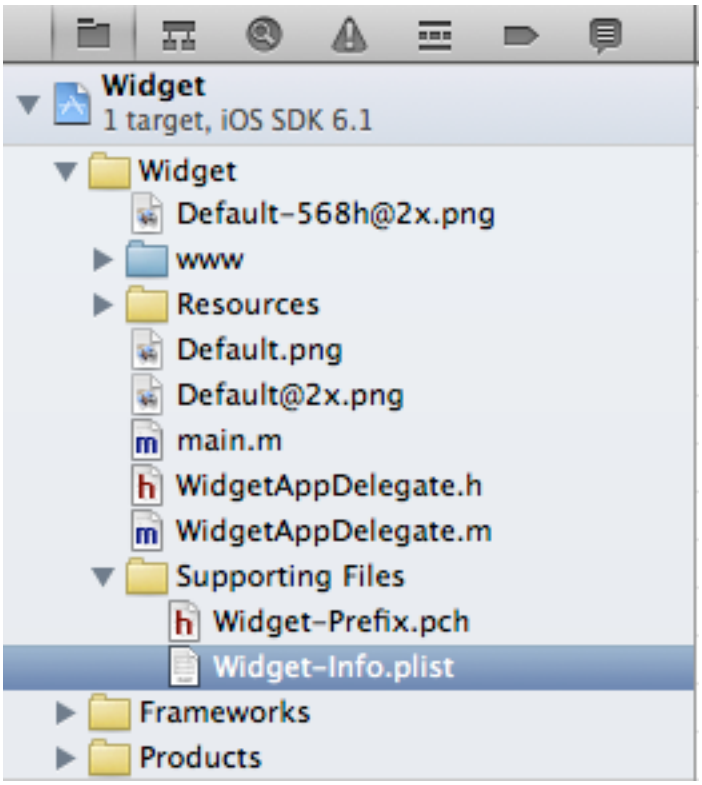
For example, to create a client called "Widget" in the ~/client directory, type:

```
./create_project.sh Widget com.acme ~/client
```


The first time this script is called, the SilverMobileCore.embeddedFramework is installed under /Users/Shared/Tibco/SilverMobile/Frameworks.

If new versions of the framework are used, be sure to delete the existing shared framework to be safe.

Once you create the project, open the iOS project in xCode. Go to the 'Supporting Files' directory and click on the 'Widget-Info.plist' file as shown below.



There are a number iOS specific properties that can be modified and created such as icons, the bundle identifier, version number, localization settings, etc. See the Apple developer site for more info on those issues. On the bottom of the page, click the 'com.tibco.SilverMobile.Customization' property to edit the Silver Mobile-specific properties as shown below.

Key	Type	Value
▼ Information Property List	Dictionary	(15 items)
Localization native development r...	String	en
Bundle display name	String	Test
Executable file	String	${EXECUTABLE_NAME}
Bundle identifier	String	com.acme.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	String	6.0
Bundle name	String	${PRODUCT_NAME}
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone envir...	Boolean	YES
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(3 items)
▶ Supported interface orientations (...	Array	(4 items)
▼ com.tibco.SilverMobile.Cust...	Dictionary	(11 items)
ShowLinksInternal	Boolean	NO
AboutURL	String	
PortalPage	String	portal.html
ServerURL	String	
ShowMessageView	Boolean	YES
ShowPortalTab	Boolean	YES
ShowRefreshButton	Boolean	YES
ShowToolbar	Boolean	YES
ShowToolbarAlways	Boolean	NO
ShowErrorDialog	Boolean	YES
StartPage	String	start.html

We will cover these items in approximate order of importance.

StartPage

As mentioned previously, the customized Silver Mobile Client can be configured in one of two ways. Like the default Silver Mobile Client in the AppStore, the first screen that appears when the app is launched points to a static, local file that uses the Silver Mobile Client JavaScript API to perform login functions. This property can point to a remote HTML page as well. It could be a single login screen or it could be a fully functioning rich app with deferred login. You can try typing any web site (like http://finance.yahoo.com, etc.) and once built and run, you'll see the yahoo finance page when the Silver Mobile Client is launched. The default start page points to the local file 'start.html'. Inspect this file to get a feeling for how the SMClient JavaScript interface works.

PortalPage

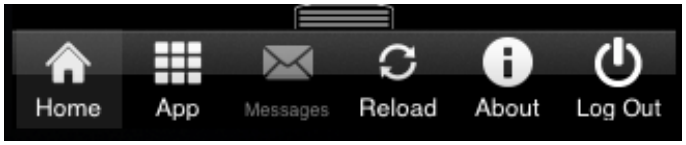
The portal view is the view that displays the list of apps that a user is allowed to access. This is typically the first screen that appears after logging in (unless the user only has permission to use one app, in which case that one will loaded automatically). The portal view can be replaced with an HTML page in order to provide individual look and feel if desired.

ServerURL

In most cases, you will not want to make the user specify the ServerURL when logging in. Use this field to set the property internally so that only the login credentials are required from the user.

ShowToolbar

The default Silver Mobile Client comes with a Toolbar (tab menu bar) as shown below:



This entire menu component can be removed from the app by setting this property to false. By doing so, however, the developer that customizes the Silver Mobile Client will need to use the JavaScript client APIs to control the login/logout logic and be able to change current app, etc if required. Some of these tabs are also removable individually as explained below.

ShowToolbarAlways

This will cause the toolbar to always be shown.

ShowLinksInternal

Any link to an external site will render within the main Silver Mobile client view. Set this flag to ‘NO’ or ‘FALSE’ to launch a separate mobile browser.

ShowRefreshButton

When you are developing the HTML mobile apps with Silver Mobile, it is convenient to have a refresh button that will clear the web cache if necessary. In a production environment, it’s likely you’d want to set the ‘ShowRefreshButton’ to false since the end-user would not need to refresh the app.

ShowMessageView

If ‘ShowToolbar’ is true, but this field is false, there will be no button for showing the message views. If any of the apps deployed to Silver Mobile want to use the default native message and message list views, it is up to the programmer to use the SMClient JavaScript interface to make the call programmatically.

ShowPortalTab

The portal tab displays the home icon and text and will take the user back to the portal view when pressed. Setting this property to false will remove this capability.

ShowErrorDialog

There are a few situations where the native client could show an error dialog (while trying to send a message offline, or if the users account was disabled, etc). This field can be set to false, but these errors should be listened to using the global error handler in the Silver Mobile JavaScript API.

6.2 Android Customization

Unzip the Android client core library to create a project directory and apply customizations. Run ant to build the project. The Android SDK (version 15 or later) and Apache-Ant (version 1.8 or later) must be installed on the development machine.

1) Expand the contents of SMC-Android-SilverMobileCore.tar.gz into a directory of your choosing to create the project directory. The directory will be named SMC-Android-SilverMobileCore. You may rename it if so desired.

2) Modify the customization properties file (found in the project directory). The available customization properties are documented in the properties file itself.

IMPORTANT: You *must* modify the sdk.dir property to point to the installed Android SDK. On a Mac, this is typically /Applications/android-sdk-mac_x86.

3) Create a resources subdirectory to contain custom logos and icons. Copy any image files into this directory. The project icon should be 72x72 pixels, and the logo image 320x48 pixels.

4) Build the project by running ant in the project directory. The customized client will be located in the build/bin directory.

7 Silver Mobile Release 2.1 IOS Project Upgrade Guide

The following are the steps required to upgrade an existing customization & rebranding IOS project from Silver Mobile Release 2.0 to 2.1. Once the steps are completed, the upgraded IOS project can deploy the rebranded Silver Mobile Release 2.1 app to IOS devices.

1) Close the existing 2.0 IOS project from Xcode.

2) Follow the steps from User Guide to expand Rel 2.1 SMC-iOS-SilverMobileCore.tar.gz to a directory <2.1 upgrade dir>. Then run <2.1 upgrade dir>/bin/create_project.sh as follows:

3) ./create_project.sh <dummy 2.1 Project folder> <com.your.bundle.prefix> <PathToCreatedProject> <path>/IOS_Framework_Rel_21 <YourCompanyName>

4) If any files under <2.0 project folder> & <2.0 project folder>/www has been customized, please make a backup for all those files in a safe place.

5) In Finder, remove the following files from Rel 2.0 IOS Project directory:
   i) <2.0 project folder>/Cordova.plist.
   ii) All files under <2.0 project folder >/www

6) In Finder, copy the following files from <dummy 2.1 Project folder> to the corresponding directories of <2.0 project folder>:
   i) config.xml
   ii) all files under <dummy 2.1 Project folder>/www to <2.0 project folder>/www

   Note: If there are customized changes made to the files that are common in Rel 2.0 & 2.1, then it is up to the user/developer to decide if those changes are needed in the migration to Rel 2.1.

7) In Finder, copy the IOS 2.1 Framework from the path specified in the (2), i.e. <path>/IOS_Framework_Rel_21 to the existing 2.0 IOS Framework folder

8) Reopen the existing 2.0 IOS project from Xcode.

9) Highlight the existing 2.0 project name folder and right click. Choose 'Add Files to "<2.0 project name>"'. It should bring up a dialog of all files in the project folder. Choose "config.xml" & click the "Add" button.

10) Cordova.plist should be highlighted in red as missing file. Right click on it and chose "Delete".

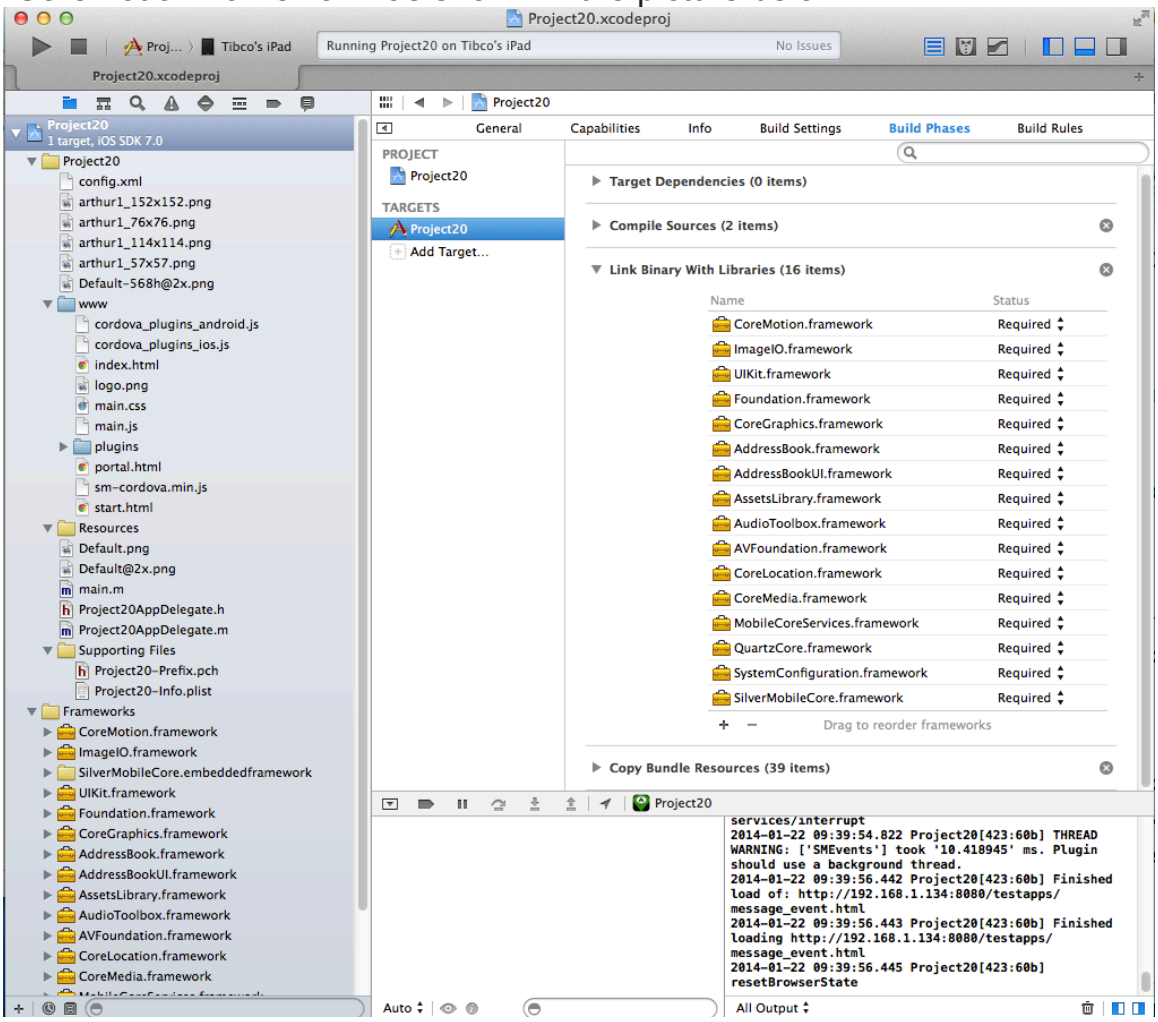
11) Click on “<2.0 project name>-info.plist and check if all the customization options are still intact. Add “portal.html” under the customization feature “Portal Page”. Also, there are 2 new customization features you can add to com.tibco.SilverMobile.Customization section: “ShowLinksInternal” (type=Boolean) and “ShowToolbarAlways” (type=Boolean) to provide further customizations to your 2.1 customization project.

12) Modifiy main.m so that it corresponds with the AppDelegate specified in the project.

13) Highlight <2.0 project name> in the navigation panel, and then highlight “<2.0 project name>” under Targets in the Project settings panel.

14) Go to the “Build Phases” tab, under “Link Binary with Libraries”, here you will see a list of linked IOS frameworks.

15) Click the “+” button at the bottom of the list and add “ImageIO.framework” & “CoreMotion.framework” as shown in the picture below.



16) In the Xcode top menu bar, click “Product->Clean”, and the “Product->Build”.

17) Make sure SilverMobile Rel 2.1 Server is installed & running.

18) Install your upgraded SilverMobile Applicaton onto your iOS devices.