WHITE PAPER

MARCH 2011

Smartphone Enterprise Application Integration

Rhomobile - Mobilize Your Enterprise

rhomobile

Overview

Smartphone app usage is exploding. This trend began with consumers adopting the iPhone and finally accepting, running and installing apps by the millions. It has continued into the enterprise with businesses using these recent, more powerful devices to make their workers more productive. One of the most interesting things about this trend, is that it is a changed paradigm in application development. For the last fifteen years application development has been mostly about centralized computing through web apps. By contrast, smartphone apps are dominated by native apps running locally on the device itself, and by utilizing device capabilities. This is truly "computing at the edge," utilizing the incredible burgeoning power of the modern smartphone. Because of this, many of the former best practices for enterprise applications need a refresh. There are **new rules for optimal smartphone app development**. The topic of this white paper is one subarea of those overall best practices: how to most effectively integrate native smartphone apps with existing enterprise applications. This white paper also discusses how **Rhomobile's RhoSync Application Integration Server**, and other components provided by Rhomobile, provide a powerful tool for enterprise application integration.

For more information on optimal smartphone development please see the Rhomobile White Paper: "**Best Practices in Enterprise Smartphone Development**"

Introduction

In the enterprise scenario there are many common tasks that need to be performed to build an informational mobile app that integrates with an existing enterprise backend app. This integration is proving to make up the majority of the effort in building these apps, and in many cases, the majority of an enterprise's entire IT efforts. This white paper will describe in detail where that effort is, and some best practices in reducing it and overall time to deployment.
When writing apps for modern smartphones to connect to enterprise backend applications, there are several areas of effort in building the full app. You need to connect to the backend application, generally through some web service protocol. You need to retrieve the payload data from that backend. The data needs to be parsed into a consumable form. Generally, the data needs to be stored locally on the device in some form of database. If the data is stored locally, then some form of ongoing database management in the face of changes needs to take place. In some scenarios the data will need to be encrypted on the device. Finally, there will generally need to be some way of making sure that the data and apps can be remotely wiped if the device is lost, stolen or the employee is terminated.

In the cases where synchronized offline data is required (which is the majority of the time for an enterprise app) many of these steps can be eliminated and the overall effort will be simplified. But, using a sync server also raises some additional steps to be performed such as: hosting the sync server, tying it in with the overall enterprise authentication directory, and writing some form of sync "source adapter" to interact with the backend application.

In talking with smartphone app developers we have found that all of these steps constitute more than half of every enterprise smartphone app's code and overall effort. In the rest of this document we will discuss each of these steps individually.

Connection

You will almost always want to connect via http or securely via https. You will need to think about how to handle authentication in a secure way. Generally, you will want to use Basic authentication challenges from smartphone devices versus some more elaborate protocol such as NTLM. On devices such as BlackBerry, you may need to explicitly control which network (Wifi or 4G carrier network for example) is used as the transport.

You should try to execute some form of asynchronous connection to keep the smartphone user interface responsive. Many desktop and web apps can get away with synchronous connections where you wait for the response. On a mobile device you will want to keep the user interface responsive while you wait for the connection and information retrieval. So some method of connecting asynchronously with a callback on the return should be performed.

Data Retrieval

The first step in integrating to the enterprise backend application is to decide how to expose that backend. The common ways of doing this now are REST and SOAP web services. You may have inherited older methods such as DCOM, Corba or XML-RPC gateways. The second step is to determine the payload format. Common approaches include XML or JSON payloads. JSON is much smaller and easier to process.

If you are communicating directly from a device, we recommend a JSON payload exposed via a REST interface. When absolutely necessary you can connect directly to a SOAP service. However, you would want to avoid this when using older BlackBerry devices, and you should be sure that the payload is not too large if you are doing this from SOAP.

When using a sync server, it may be OK to use some legacy method as the sync server can act as a proxying mechanism to make data easily retrievable by the device.

Parsing Responses

Once you retrieve the data, you will then need to extract the data that your app needs from the payload. The options for doing this are most often either JSON parsing or XML parsing. If you are handling a large XML payload you will want to consider using either a stream or pull parser versus using the XML Document Object Model (DOM) to retrieve the data.

If you have a large amount of data (greater than 10MB) you will want to consider using some more efficient method than a text stream to get the data to the device. You could create some archived compressed format and deliver that in bulk to the client device (a technique sometimes referred to as “bulk sync”).

Populating the Database

Once you have the data, generally you will want to provide some way of populating a database or local cache. Users should not have to wait each time they start their app for their data to be retrieved from the server. Generally this means taking the parsed data and creating records in a database. Also, in order to have users be comfortable creating or changing data for an enterprise application, you will need to give them a way to store their changes locally before it is applied. This requires managing the local database. It also generally requires some approach to data synchronization, which we will discuss shortly.

Secure Storage

In some cases involving local data, you will want to encrypt that data before storage. This could be because of regulatory standards such as HIPAA, or the data could contain sensitive information such as personal financial information or confidential trade secrets. Because there is a chance of the device being stolen or lost, it is important to securely encrypt such sensitive information. However, be careful when applying this technique as encryption tends to be slow especially on older BlackBerry or Windows Mobile devices. Use it where necessary and nowhere else.

App and Data Deprovisioning

In the cases where sensitive data is on the device, (whether or not the data is encrypted) you will want to have a way to deprovision apps and data if the device is lost, or an employee is terminated. The app and data management capability should have the ability to utilize the user list from the enterprise directory. There are many mobile app and device management solutions from vendors such as Sybase, Good Technology, Mobile Iron and Rhomobile. You should strongly consider having one of these solutions in place as part of your general smartphone app planning, especially when considering synchronized data (highly recommended as described below).

Rhomobile is the only cross platform mobile framework that is Enterprise Security Compliant. Check out our White Paper "**Security in the Mobile Enterprise**" to learn how your mobile strategy is secured from end-to-end with data encryption, remote device wipe, and secure authentication

Data Synchronization

As mentioned above, in order for end users to utilize your app in enterprise application usage scenarios, you will need to provide some way for users to store their changes locally, whether or not they are connected. This is important for several reasons. First, the most obvious reason is so that users can have their data and do their work (such as creating and editing information) when they are offline. A more critical reason for offline data synchronization is so that users know that their changes are made reliably, once and only once. Users send and receive email with rich client apps on their mobile devices – apps that store data locally and sync the email in the background (versus using a mobile web browser). They have the same expectations for their enterprise apps. Finally, having the data available locally creates a faster user experience – the data is always available for them immediately on startup.

A side benefit of using data synchronization is that it simplifies the app integration scenario described above. Instead of doing all the work to connect, retrieve, parse and store the data, a good synchronization server and client app framework will perform all of that work for you. The only task for the app developer is to write the code for the sync server to connect to the backend application. That work is typically called a “source adapter”. The app developer needs to write code to enable the server to retrieve or query information. If he wants bidirectional data he will need to write code to create, update and delete information as well. This is typically much less code than performing all of the connection, retrieval, extraction and storage from the device. And, usually, the sync server has a richer set of libraries and tools available to make those backend connections. The server can usually handle whatever format the backend application exposes, versus requiring a new web service to be exposed.

You should ensure that whatever server you use for synchronization supports “push sync”. This means that changes made to the backend application get immediately sent to the devices that need them. This avoids the device having to “poll” occasionally for changes. Polling results in mobile device battery drain and some level of staleness of data for the user. Note that your backend application must have some kind of notification mechanism exposed. Also, there must be some way of performing a push to the device to make this work. It is unusual that “homegrown” approaches to sync do push.

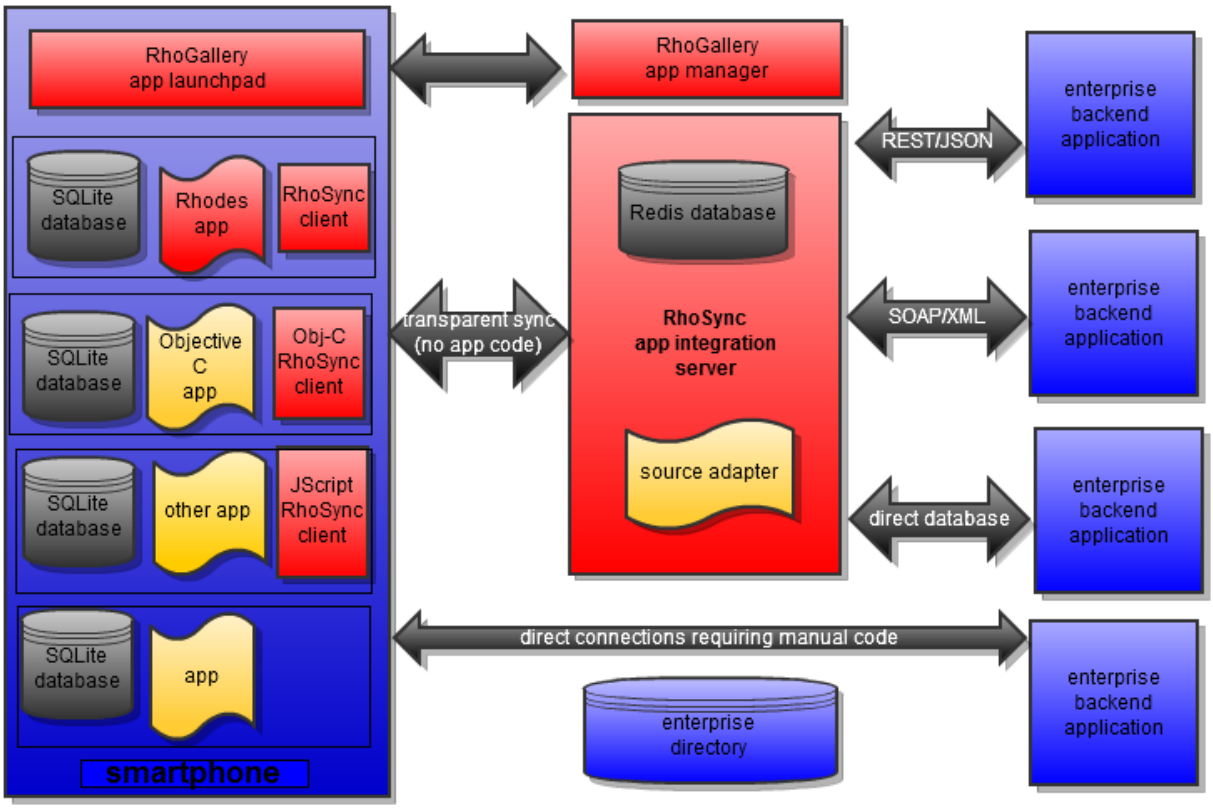
How Rhomobile Can Help

So the major tasks in writing mobile apps to integrate with backend applications are: connecting to backend apps, retrieving the data, parsing responses, populating the local database, secure storage, app and data deprovisioning, and data synchronization. In discussions with smartphone app developers writing apps in Objective C or Java, or using frameworks like Rhodes or PhoneGap, we usually hear that these tasks involve the majority of the development effort and code size.

The rest of this white paper describes how the **RhoSync Application Integration Server**, and other components provided by Rhomobile, helps with each of these tasks, drastically reducing the effort involved in mobilizing an enterprise application.

RhoSync automates most of this process by allowing the developer to just provide a “source adapter” for each model. The RhoSync server then does all of the work to connect to the backend application, retrieve the data, parse the response, and populate a server cache (specifically a very efficient Redis database) with the information. The RhoSync server then monitors all of the smartphone devices that connect to it and efficiently sends them just changes to the information that they need. The RhoSync server can be used whether or not the smartphone app was written with the Rhodes framework. Rhomobile products ship with Objective C RhoSync client. A JavaScript RhoSync client is under development.

Rhomobile Architecture



Writing a RhoSync Source Adapter

Note that in most cases a “source adapter” must still be written to connect to the backend application. Specifically, this means writing “query”, “create”, “update” and “delete” methods to interact with the backend system. The source adapter query method returns data in a standard “hash of hashes” result and the RhoSync server does the rest of the work to get all of the data down to the device’s database. The create, update, and delete methods which you write can expect object attribute and value information to be provided to it by the RhoSync server as a result of whatever the client app has done to create or update record information. For example, the query method might connect to the backend application’s REST interface, retrieve the data as a JSON payload, and then put the data into a standard @result "hash of hashes" variable. This saves huge amounts of code versus the typical "direct connect" way of handling data feeds. The code can be as simple as:

```
def query
      items=JSON.parse(open(“http://yourcompany.com/
      crm/accounts.json”).read)
      @result={}
      items.each do |item|
            @result[item["id"]]=item
      end
end
```

Assuming an app that might just retrieve information, those five lines of code replace hundreds of lines of Objective C or Java code of an app that might connect directly. And of course, you get the benefit of automatically synchronized offline data in the process.

Bulk Sync

With the creation of a source adapter, RhoSync handles the connection and retrieval of data, parsing the response and populating the local database. Note that RhoSync also handles performing large database delivery through its “bulk sync” feature. If bulk sync is turned on the RhoSync server will create a SQLite database on the server and send it down to all devices compressed. This is very helpful for models (business objects) with large amounts of data such as product catalogs or customer account lists.

Rhodes Automatic Encryption

For sensitive data (such as patient information in a medical information system or personal financial information in a banking system), the enterprise will often want to have the data be stored encrypted on the device. If the smartphone app is written with Rhodes and the “secure app” option is turned on, then the data will be populated in encrypted form into the database. No code to perform encryption needs to be done in the smartphone app.

Automated Deprovisioning

Another concern when integrating smartphone apps with critical enterprise systems is what happens if a device is lost or stolen or an employee is terminated. RhoGallery allows apps and data to be deprovisioned whenever the IT administrator chooses. RhoGallery is an app management feature on the **RhoHub** hosted development site that also provides a client app that can be used to control app execution. RhoGallery integrates with the enterprise directory (such as LDAP) to get the list of acceptable users and determine if users are still valid.

RhoGallery is the first hosted mobile app management solution. RhoGallery allows companies to easily manage and provision apps for all of their employees regardless of device type. Visit **www.RhoHub.com** to learn more.

Synchronization

Primarily, this paper has described how RhoSync eases the app integration process versus writing code directly from the device to integrate with backends. But of course, RhoSync also provides for synchronized offline data. This means that users can fully use their apps whether or not they are connected to the Internet from their device at a given time. It also means a higher perceived level of performance for the user. The data is always present on the device. The user does not have to wait for information to be retrieved from the backend.

Note that writing an app that just retrieves information and stores it on the device is not synchronized offline data. One reason is the lack of “push data”; data must be retrieved by explicit request or action by the user. With RhoSync data can be pushed to the device realtime (it is unusual to unheard of for “manually written direct from device to backend application sync” to do push data). Another reason is that without a server like RhoSync to keep track of the “master list of data”, the device cannot receive only the changes.

“No Code” Integration for Ruby on Rails Developers

If you are writing your backend application in Ruby on Rails we will soon be providing a Rails plugin that literally removes all of the code necessary to synchronize the data to the smartphone app on the device. In other words, a source adapter is no longer necessary. By including the “RhoSync Rails plugin” in your Rails app, your Rails app will push data through RhoSync directly to the device. Data associated with all models (or selected models) of your Rails app is replicated efficiently down to the device. No source adapter needs to be provided at all. Note that for most cases a RhoSync source adapter still needs to be written, but for Rails developers there is a unique opportunity to completely eliminate code for app integration.

Summary

If you choose to use RhoSync it should alleviate most of the effort described in this paper. It automates much of the repetitive and laborious process involved in integrating with backend data feeds. In some cases RhoSync can remove literally all of the code required to integrate with a backend app. Use of Rhodes to write your app makes encryption of data automatic where appropriate. Using RhoGallery to manage your apps handles the app deprovisioning and management issues with enterprise apps and data.

Regardless of whether or not you use Rhomobile products, if you are considering building an enterprise app for modern smartphones, we would encourage you to look at each of these steps to performing integration with your backend enterprise application. Thinking carefully through each of these issues is likely to accelerate your development efforts and result in a more performant, maintainable, and secure solution.

Rhomobile
Mobilize Your Enterprise

Rhomobile, Inc.
3031 Tisch Way Ste. 704
San Jose, CA 95128
877.RHO.0334
408.572.8076

www.rhomobile.com
www.rhohub.com

About Rhomobile

Rhomobile's free and open source mobile application framework, Rhodes, lets you quickly build native mobile applications for all smartphones: iPhone, BlackBerry, Windows Mobile, Symbian and Android. These are true native device applications (not mobile web apps) which work with synchronized local data and take advantage of device capabilities such as GPS, PIM contacts, camera, native mapping, push, alerts and calendar. Rhodes invented the smartphone app framework (including that term) two years ago and has had over 50,000 open source downloads since then. Rhodes is the only smartphone framework with support for all smartphones, a Model View Controller pattern, and, most importantly, support for synchronized offline data. Because of these features, which are all critical for enterprise needs, several large Fortune 500 companies have standardized on Rhodes for mobile development.