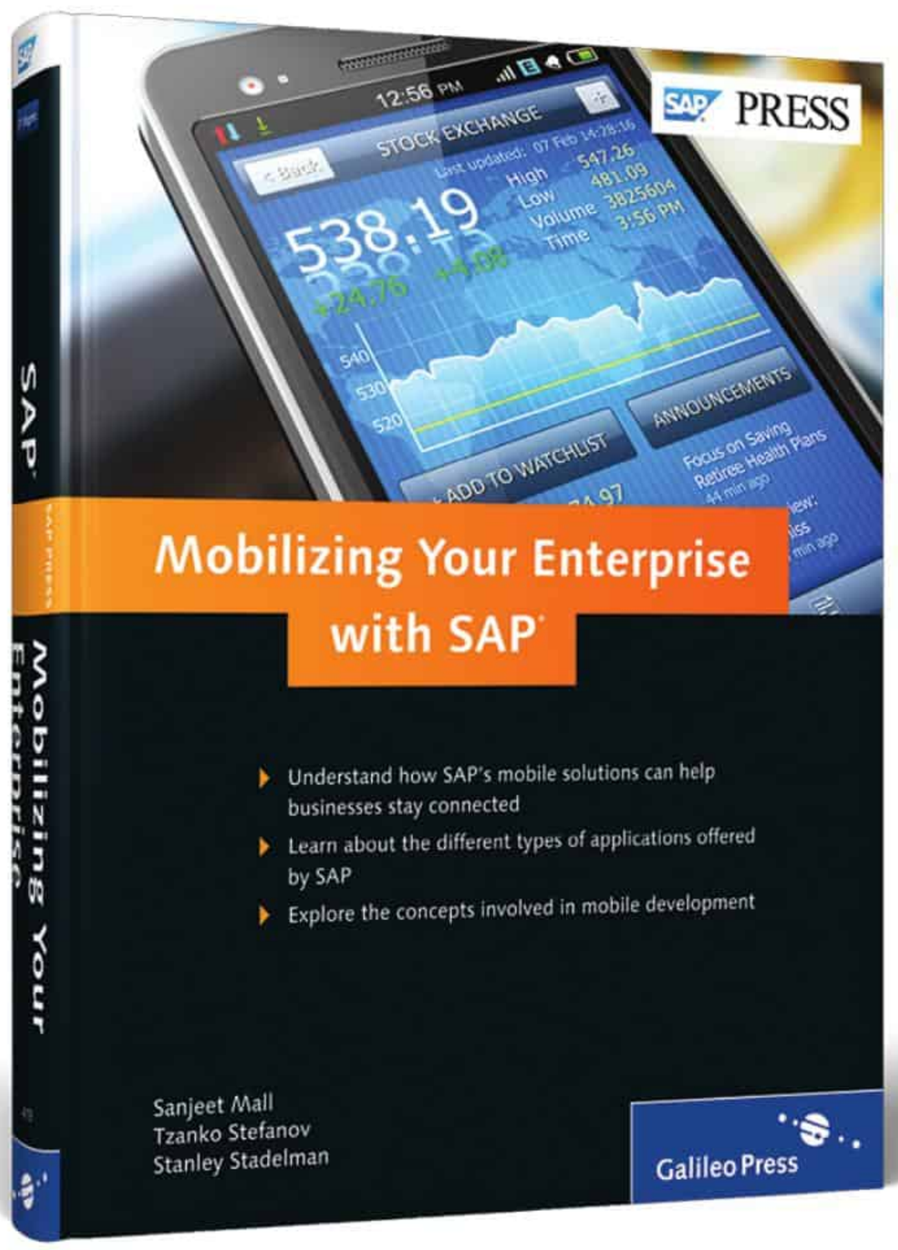
Sanjeet Mall, Tzanko Stefanov, Stanley Stadelman

Mobilizing Your Enterprise with SAP®



Galileo Press

Bonn • Boston

Contents at a Glance

Contents

Now that you know the basics of mobility, learn how SAP's tools fit in with a mobile strategy.

3 Introduction to the SAP Mobile Platform

Walldorf, Germany, the heart of SAP and where it all began, is a small town located along the A5 motorway midway between Frankfurt and Stuttgart. Rumor has it that there was a time when the IKEA (at that time, the biggest thing in Walldorf) was the point of reference for giving directions in town. Now, of course, Walldorf is synonymous with SAP.

The SAP journey has been a long one, and while we won't outline the full history here, it's a journey that has been very much aligned with the four waves of computing (discussed in Chapter 1). Here, we will recall Figure 1.1 from Chapter 1, which showed these four waves, but this time superimpose the figure with the key technologies and products that powered SAP's growth from a small shop to one of the best known brands in the world (Figure 3.1).

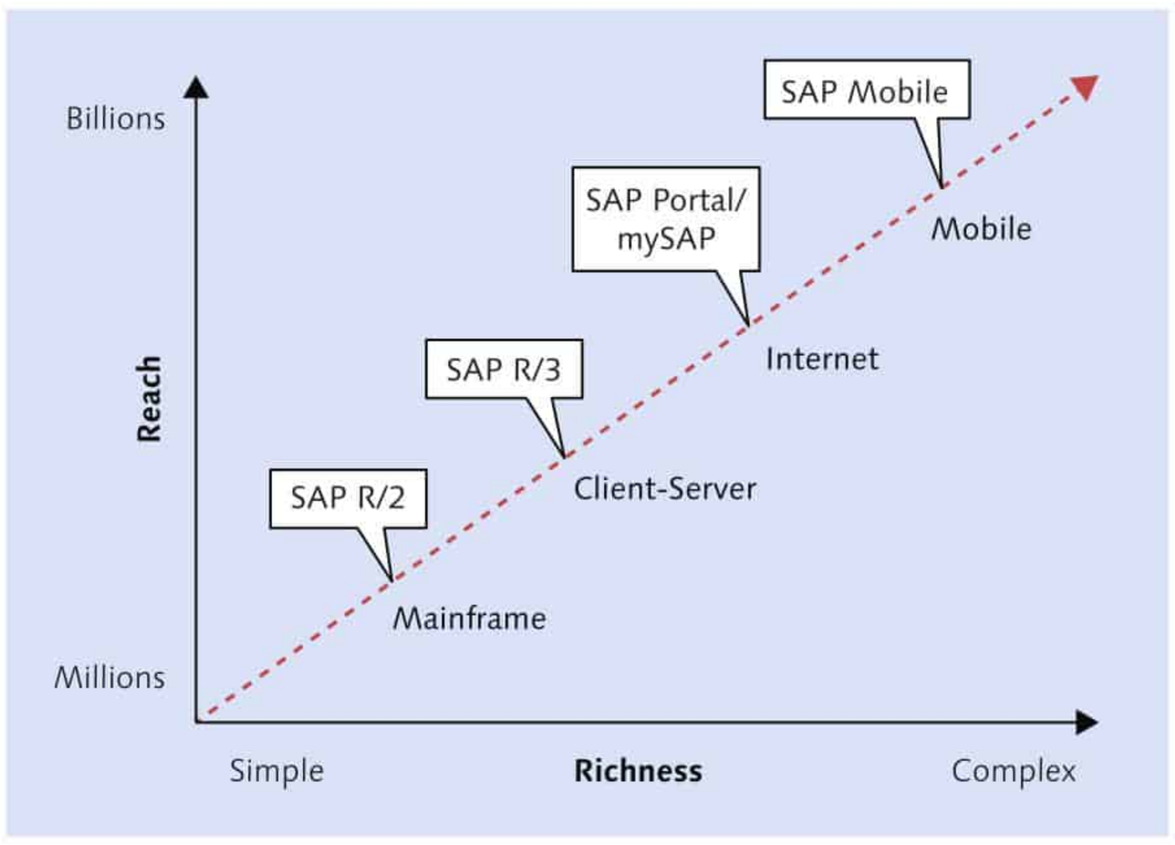


Figure 3.1 The Four Waves of Computing and SAP

In 1980, SAP introduced R/2, which succeeded SAP's financial accounting system known as R/1. R/2 was based on a two-tier architecture ("R" stands for real-time, and "2" is a reference to this two-tier architecture). It ran on mainframe computers and integrated many enterprise functions such as accounting, manufacturing, logistics, supply chain, and human resources.

R/3 was introduced in 1992. As the name suggests, it was based on a three-tier client-server architecture that separates the presentation layer, application layer, and database layer into distinct subsystems. R/3 was divided into the following distinct modules: Sales and Distribution (SD), Human Resources (HR), Product Planning (PP), Materials Management (MM), and Financials and Controlling (FI/CO). Each module handled its business task on its own but communicated with the other modules when required. For example, an invoice from SD would pass through to the FI module as accounts receivable.

Keeping pace with the Internet age, SAP's mySAP suite of products—CRM (Customer Relationship Management), ERP (Enterprise Resource Planning), SRM (Supplier Relationship Management), and SCM (Supply Chain Management)—made their functionality available (by using HTML to render the user interface [UI]) through a browser. An end user just needed a browser to get to the needed functionality to complete a process. The process itself could span multiple systems, but, to the end user, it looked as if he was just working with a single system with a browser as the frontend. Central to this architecture was the concept of role-based access. Depending on the function you needed to perform in your company, the mySAP suite of applications exposed functionality from multiple systems in a cohesive manner in one central place—the mySAP Workplace. This workplace was your entry point to all of the information you needed to complete your job. SAP NetWeaver Portal was the glue that bound it all together: managing and hosting the mySAP Workplace, role creation and assignment, managing single sign-on (SSO) (so the user just logs in once and is granted access to all the systems required for that role).

SAP NetWeaver Portal took SAP products into the Web 2.0 era by providing features such as collaboration, personalization, and content management (management of unstructured content). SAP NetWeaver Portal also opened the doors for SAP applications to work in mashups with content outside (Internet, third-party applications) of native SAP applications in a flexible (loosely coupled) way that

increased the net value of the information. For example, account information can come for SAP CRM and a map location of the account from Google all in a single SAP NetWeaver Portal screen. In another example of the value of mashups, order and shipping information is presented from SAP ERP and delivery tracking from the carrier (DHL, UPS, etc.) to give a complete overview of the order fulfillment status in a single screen hosted by SAP NetWeaver Portal.

Mobility offerings aren't brand new to the basic SAP landscape. SAP has had a number of mobile products such as SAP CRM Sales Laptop, Mobile DSD (Direct Store Delivery), and other focused solutions that solved specific problems. In other words, the last generation of mobile in SAP (prior to the Sybase acquisition) was extending the SAP Business Suite to mobile solutions. This had limited scope, however, because it was not meant or designed to revolutionize SAP for the brave new world of mobile.

The acquisition of Sybase in 2010 told a different story, however. First, it indicated that SAP was extremely serious about the mobile revolution; and second, it meant that SAP now had a mature set of tools covering every aspect of mobile enablement (a mobile enterprise application platform [MEAP], mobile device management [MDM] tools, etc.), which gave customers what they needed to kick-start their own mobile revolution. With a mature mobile platform and a full complement of mobile applications coupled with explosive growth in mobile adoption and rapid improvements in capabilities of services and devices, SAP is extremely well placed to make the most out of this wave of computing.

This chapter provides an overview of the key elements of the SAP Mobile Platform; specifically: Sybase Unwired Platform, SAP Afaria, and the Sybase 365 messaging and Mobiliser mobile commerce products. Combined with SAP NetWeaver Gateway, these technologies provide the core of the SAP Mobile Platform. (For more on SAP NetWeaver Gateway, see Part III: SAP Mobile Development.)

3.1 Introduction

Sybase has been a leader in mobile technology since early in the mobile laptop and smartphone era. It acquired Waterloo-based iAnywhere in 2000 for the embedded database SQL Anywhere, which led the mobile database market in

2001 for the fifth consecutive year.[1] In 2003, Sybase acquired AvantGo, an offline-enabled browser for PDAs; in 2004, it added XcelleNet, a frontline device-management software provider; and in 2005, it acquired Extended Systems, a provider of technology used in the Sybase OneBridge Mobile Office email solution. These technologies enabled development and management of Windows- and Java-based applications, primarily for cases in which devices would be occasionally disconnected, and the data and transactions in the mobile application were replicated from a JDBC data source, using a technology component named MobiLink. Developers could configure the behavior of the MobiLink synchronization using MLScript, a proprietary scripting language, and SQL.

By 2007, it became clear that smartphones would become a significant application platform in all enterprise IT organizations, and Sybase developed a mobile platform Software Development Kit (SDK) to simplify and standardize mobile application development for the leading mobile devices. Sybase Unwired Platform mobile platform was the solution for unwiring the enterprise. Sybase Unwired Platform enables enterprises to develop, deploy, and manage mobile applications across leading smartphone and tablet device platforms. In Q2 2012, SAP included Sybase Unwired Platform in the SAP Mobile Platform product bundle (Figure 3.2), which also ships with SAP Afaria, SAP NetWeaver Gateway, SAP NetWeaver Mobile, and the Mobiliser application server.

The SAP Mobile Platform enables application development for a complete spectrum of enterprise application types:

- Process-centric applications
- Productivity applications
- Consumer applications
- Analytic applications

Note

For more detailed descriptions of these different types of applications, refer back to Chapter 2.

1 Source: Gartner, DataQuest: *www.thefreelibrary.com/iAnywhere+Solutions+Leads+Mobile+Database+Market+For+Fifth...-a077222054*

The SAP Mobile Platform provides a set of server-side and client-side SDK components for native applications, hybrid applications, and open HTTP client applications. The product ships with Eclipse-based tooling for modeling data services and managing synchronization behavior in process applications, as well as a workflow designer for hybrid applications. Developers use their Integrated Development Environment (IDE) of choice for client application development, including xCode for iOS, Eclipse for Android and BlackBerry Java, and Visual Studio for C#/.NET development; Adobe Dreamweaver, Sencha Architect, or others for HTML5 and JavaScript development; and Titanium IDE for Appcelerator development.

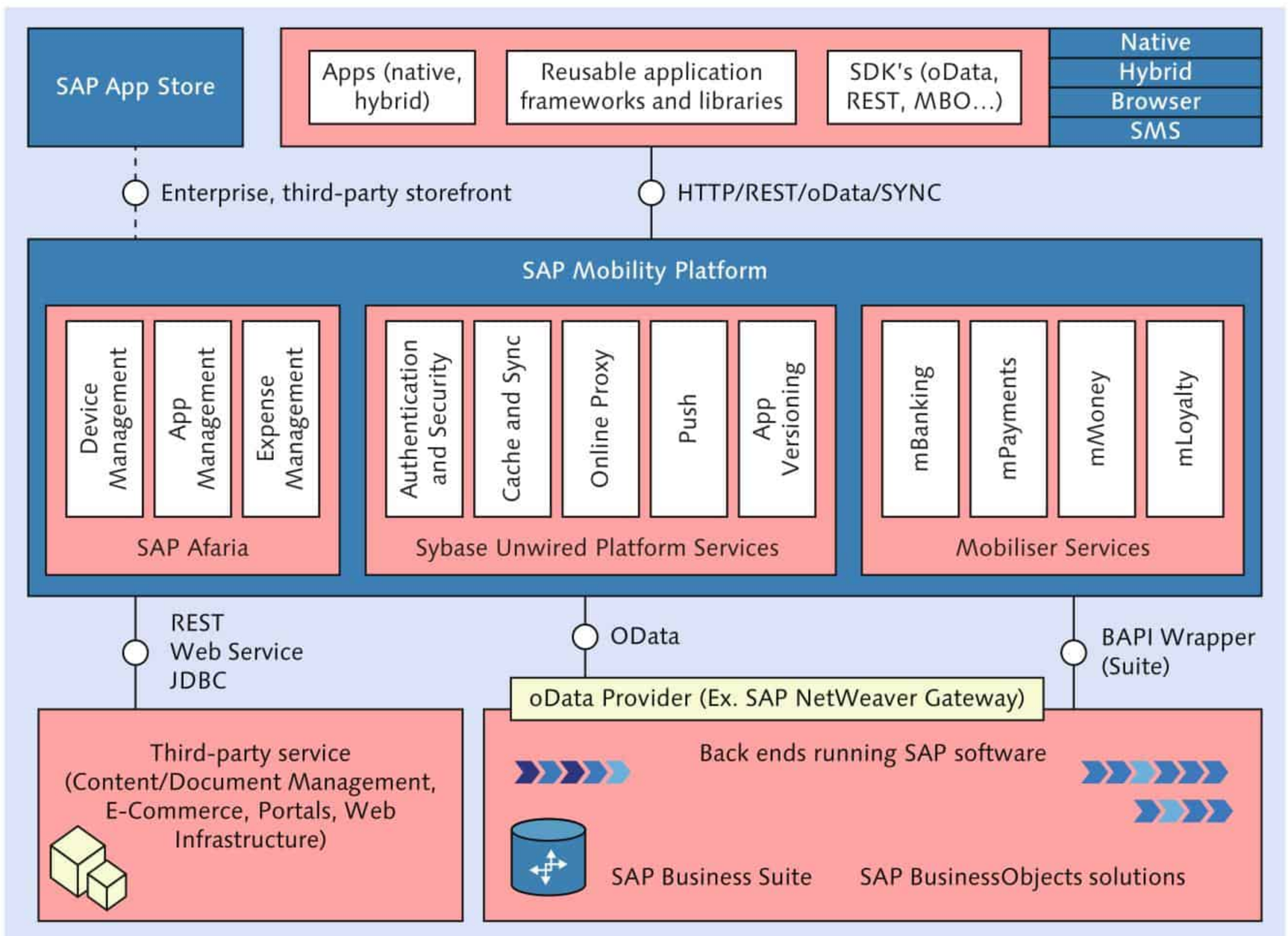


Figure 3.2 SAP Mobile Platform

The SAP Mobile Platform provides the following mobile platform services that can be integrated into mobile applications to enable simple integration into existing enterprise IT landscapes:

- Application registration
- User authentication and SSO management
- Registration for push notifications
- Settings and configuration management
- Administrative reporting
- HTTP service proxy and whitelisting

Additional features include client-side data security, libraries for developing with OData web services, rule-based data distribution, application modules for mobile payment and banking transactional services, automated application bootstrapping, and role-based application distribution.

3.2 Sybase Unwired Platform 1.x

In 2008, Sybase began a project to develop an industry-leading platform for building, deploying, and managing mobile applications in a way that is simple, fast, cheap, flexible, and open. The mobile platform would provide a set of comprehensive services to allow customers to mobilize appropriate data and business processes for enterprises using any mobile device. The platform would include the following:

- 4GL (fourth generation language) tooling and integration with standard development environments
- Deployment to multiple devices (Windows Mobile, RIM, etc.)
- Integrated device management and security throughout

The mobile platform would reduce overall costs of development, deployment, and management of these applications by enabling a design-once, deploy-to-many approach to mobile application development. In addition to 4GL data source integration, it would also include 4GL device application development, while leveraging existing developer expertise with support for Visual Studio and Eclipse development platforms. It would also support out-of-the-box integration with enterprise applications such as SAP and Remedy, as well as a full web services stack.

Sybase Unwired Platform version 1.0 (Figure 3.3) shipped in Q3 2009, with support for Windows XP, Windows Mobile 2003 Pocket PC, Windows Mobile 5.0

Pocket PC, Windows CE 5.0, Windows Mobile 6 Standard and Professional editions; .NET 2.0, 3.0, and 3.5; .NET Compact 2.0 and 3.5; and RIM Handheld version 4.2. Sybase Unwired Platform introduced an object-oriented (OO) paradigm on top of mobile middleware based on the core SQL Anywhere and MobiLink technologies called the *Object API* (Application Programming Interface) for application development, and it integrated a small set of device/security and provisioning management capabilities. The Object API development SDK is based on a core concept called the *mobile business object (MBO)*. Next we'll discuss MBOs in more detail, as well as the most important features of Sybase Unwired Platform 1.x: Object API code generation, the Device Application Designer, and the Mobile Workflow application.

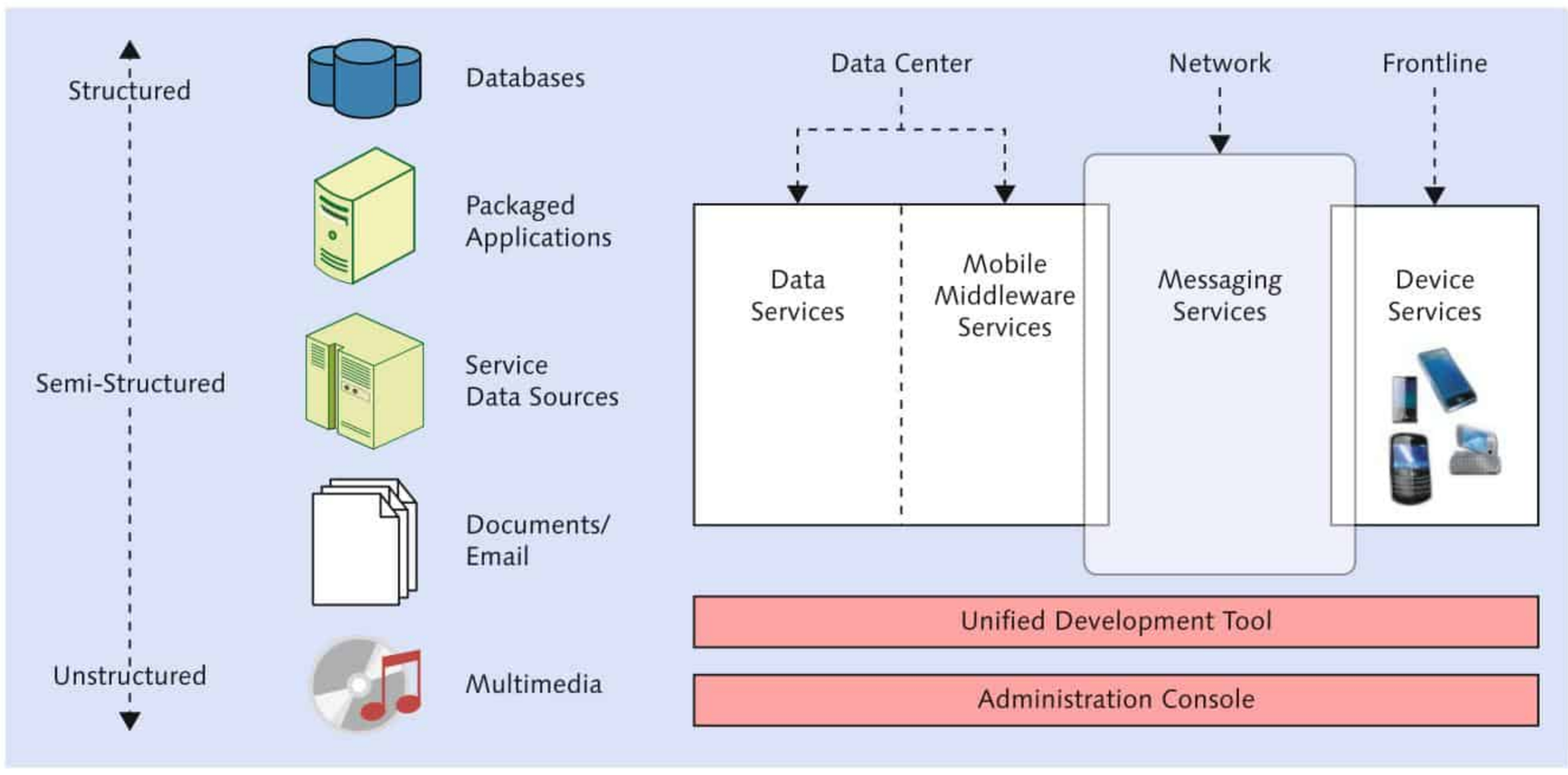


Figure 3.3 Sybase Unwired Platform, Version 1.x

3.2.1 Mobile Business Objects

The mobile business object (MBO) provides an end-to-end solution for connecting to multiple types of backend systems—including SAP—modeling the data that will be exposed to the mobile devices, and then generating code for the developer to use in the application. As such, MBOs are a core concept in the Sybase Unwired Platform related to synchronization and client-side development. Sybase Unwired Platform includes an Eclipse-based plugin for developers

to use that allows them to connect to backend data sources. They then graphically model the connections as MBOs.

The MBO is a data definition characterized by the following properties:

- Backend data connection (data source)
- Attributes
- Operations
- Relationships
- Query parameters

The tool allows the developer to see the definitions of the data sources to which he's connected, and drag and drop the data source into a MOBILE APPLICATION DIAGRAM canvas. Figure 3.4 shows this action, in which the data table is dragged from the ECLIPSE ENTERPRISE EXPLORER window to the MOBILE APPLICATION DIAGRAM canvas window, creating an MBO. After a number of MBOs have been created, the developer can drag between them to create relationships.

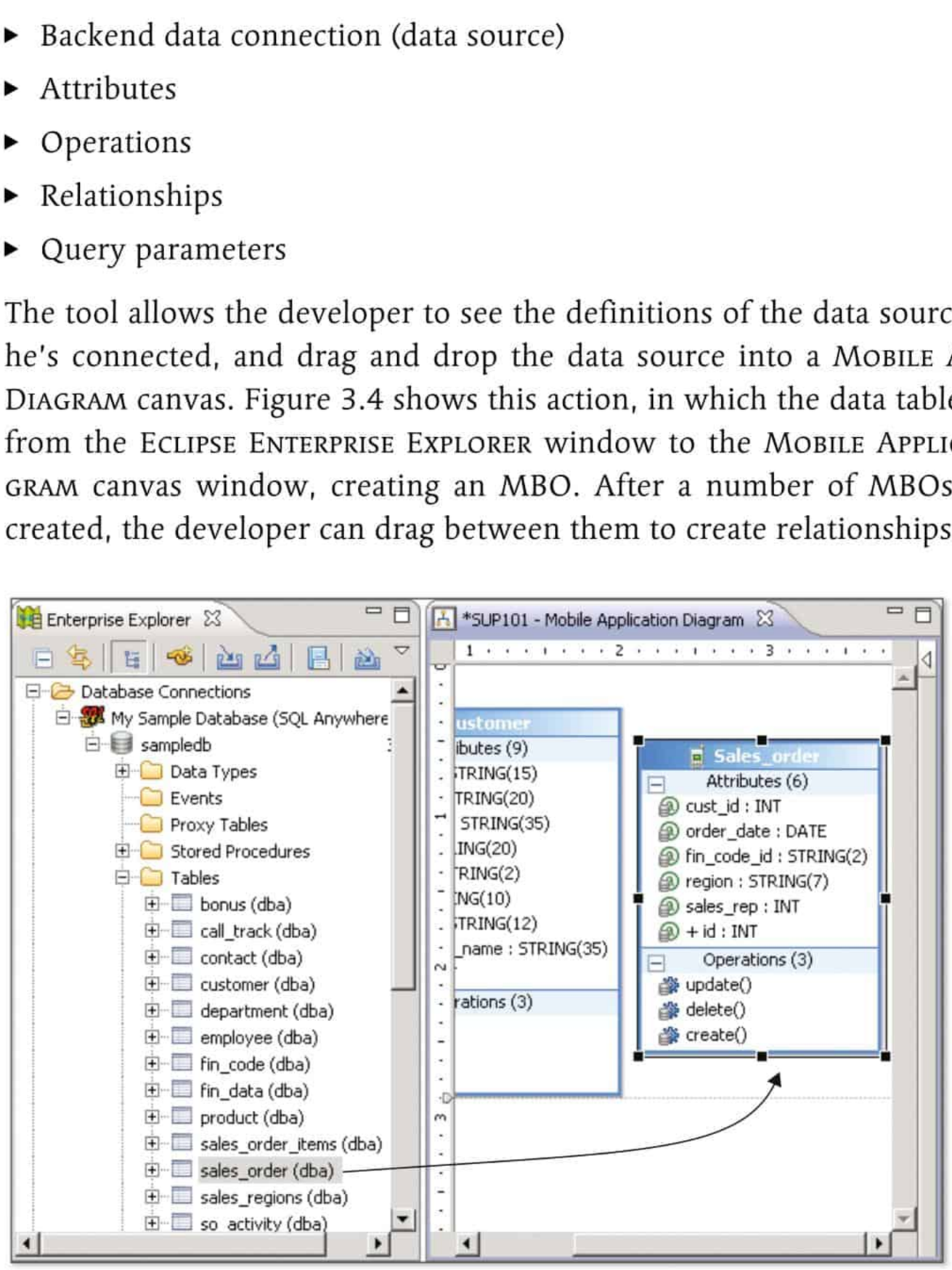


Figure 3.4 Sybase Unwired Platform Eclipse Tooling

When the developer is done modeling, the model on the canvas resembles a relational database model that you might see in a typical 4GL tool. One of the key

steps during the MBO modeling process is determining exactly which attributes that are available in the data service will be accessible on the mobile device. This is typically a small subset, (e.g., 20 attributes of 180 available in the BAPI), and it's important to optimize the MBO because it defines the data that will be synchronized to the device. Reducing the content of the MBO to the minimal requirements results in superior application performance because less bandwidth is required to sync updates to the device, and queries to the local database on the device will be faster. The tool provides a wizard for configuring this optimization in the MBO and can expose the input and output attributes from the data source to the developer with a simple checkbox to determine what is exposed to the mobile application and what isn't. For web services, the developer can also modify the definition of the data connection via an Extensible Stylesheet Language Transformations (XLST) document and map the MBO attributes to the custom document.

After modeling the MBO attributes and relationships, the developer should configure the settings for how often the data is synchronized between the backend and device, and vice versa. These settings can be configured for every MBO *individually* or for *groups* of MBOs. This enables the developer to take into account how frequently the specific MBO data must be refreshed to the device to optimize performance. For example, if an MBO is linked to the product catalog, which is only updated weekly, it doesn't make sense to keep re-refreshing the catalog data to the device every hour or every day. The product catalog MBO can be set to sync weekly. In the same application, the account activity is updated in nearly real time. The account activity MBO can be set to refresh every 5 or 15 minutes, and the user can click to refresh if the application is open. Both of these MBOs coexist in the same application, use the same database on the client, and have the same object classes generated by the tool for the developer, but they can be configured to optimize performance based on their specific requirements.

The synchronization is a two-step process designed for minimizing load on the backend application while enabling the periodic or on-demand refresh requests from the device. Figure 3.5 shows this two-step process. The first step is the loading of data from the backend application data source to the Sybase Unwired Platform server, where it's written to the cache consolidated database (CDB). Data is loaded into the CDB either by a push notification from the backend application (data change notification [DCN]) or via a request from the CDB. These settings (for loading the cache) are called the *cache refresh settings* for an MBO, or—if there

is a group of common or linked MBOs—the *cache group*. The CDB doesn't just completely reload itself every time a device is set to sync. It loads only the data that is relevant for the devices that are synching, and then it partitions itself internally to be able to determine which requests from devices are identical, so that if they share security configurations, the result set on the CDB can be shared. This results in very efficient loading and refreshing of the CDB data based on the MBO settings and the requests directly from users on the device.

Figure 3.5 Synchronization Stages in the Sybase Unwired Platform

The second step of synchronization is between the CDB on the Sybase Unwired Platform server and the devices. In the same way that MBOs have cache (group) settings for how often the cache is refreshed from the backend, MBOs have synchronization (group) settings for how often the device database is refreshed from the CDB. Typically, a sync group will be coordinated with a cache group, but there are times in which a sync group will map to a number of smaller cache groups, which are broken up to reduce the time required to refresh from the backend into the cache.

After the MBO model is designed, and the MBO settings are configured, the developer deploys the package to the server. This step sets up the CDB to accommodate the model and prepares the Sybase Unwired Platform server to have devices start connecting to it.

Note

MBOs are a core concept of Sybase Unwired Platform, which is why we briefly mention them here. However, they will be covered in much greater detail in Part III: SAP Mobile Development.

3.2.2 Object API Code Generation

After configuring and deploying the MBO definition, the developer generates the client-side code for the data model. This consists of a set of native object classes—one for each MBO—a set of helper object classes for interacting directly with the client-side database, and a set of libraries for creating the database, managing the connection to the server, and handling other communications.

The Object API code is native code: C#, and Java for BlackBerry devices, Objective C for iOS (added in version 1.5), and Java for Android (added in version 2.1 ESD#1). The developer links the code into a new mobile application project in the native IDE for the target platform (Visual Studio or Eclipse), or the developer uses the Device Application Designer to develop a client-side interface and business logic for a cross-platform application.

Note

The Visual Studio plugin for mobile application diagram development was deprecated before version 1.5.2 of the product.

Object API code may be modified by the developer, but changes aren't reverse-compatible with the mobile application diagram, so if modifications to the diagram are made, the code must be re-generated, and changes made directly to MBO or database classes will be overwritten.

Note

Device Application Designer was deprecated before version 2.0 of the product. Developers using versions 2.0 or later will develop native code in the typical IDE for a target platform or integrate with third-party tools.

3.2.3 Device Application Designer (Deprecated)

The Device Application Designer (Figure 3.6) is a 4GL plugin in Visual Studio and Eclipse development environments for designing client-side native applications with screens that are bound to MBO content and create, update, and delete (CUD) operations. The tool provides a palette with a set of stock screens, and the developer can design the screen flow of the application, bind to data definitions, create actions, and access event properties.

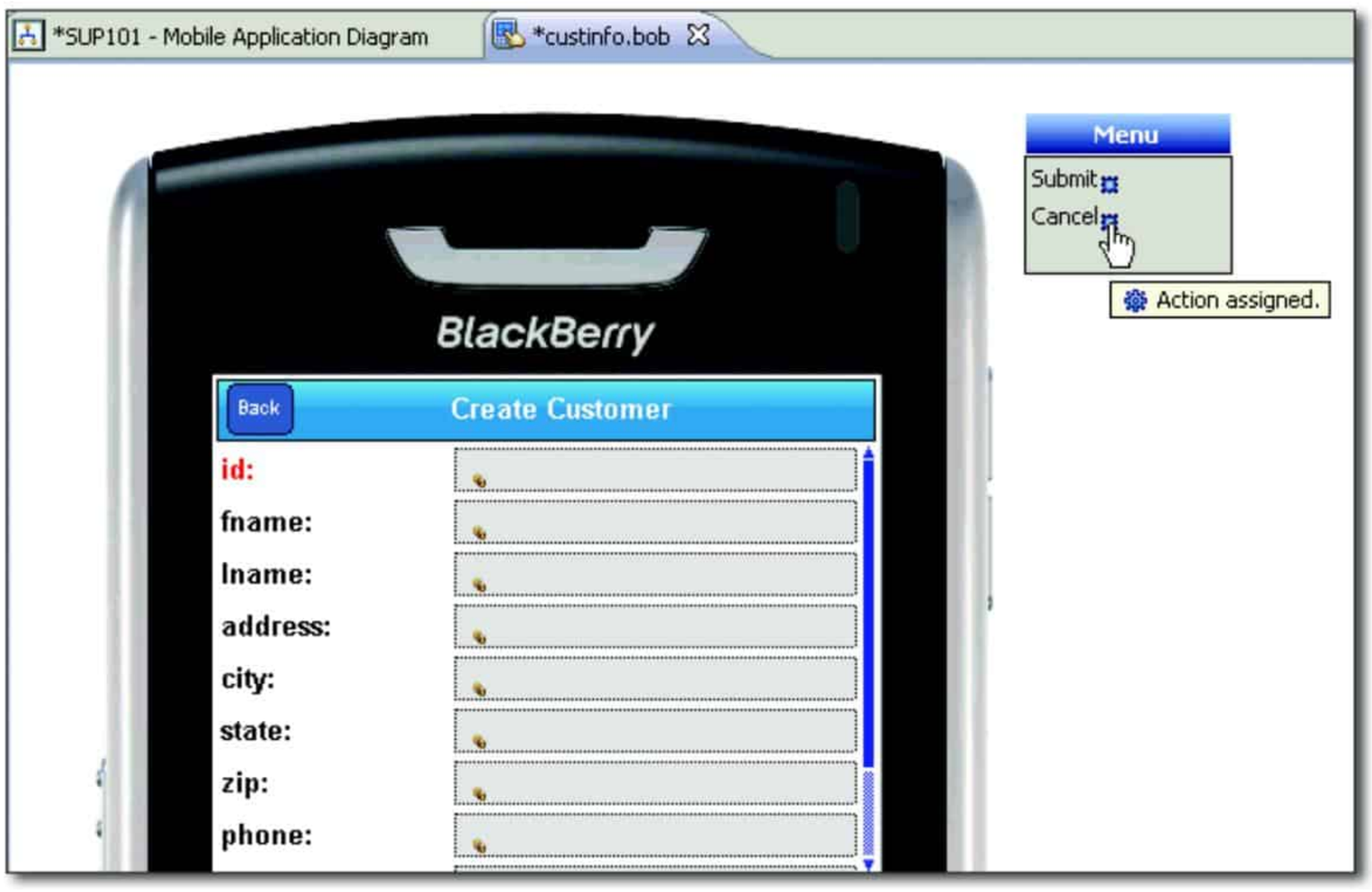


Figure 3.6 The Device Application Designer Tool

The developer selects from a set of device skins for BlackBerry and Windows Mobile devices to configure the screen, according to the size and ratio. The tool deploys the prototype application to the device simulators provided by Microsoft and RIM.

Note

The Device Application Designer was deprecated before version 2.0 of the product. The Eclipse version of the tool for generating BlackBerry client applications was posted to the SAP SCN as a community download for public access at *https://cw.sdn.sap.com/cw/groups/sup-apps*.

3.2.4 Mobile Workflow Application

The Mobile Workflow Application is a tool that was added to the client SDK in version 1.5.2 of Sybase Unwired Platform. It is a native application that is installed on an end user's device and connects to the Sybase Unwired Platform server. An application package, including an XML document describing the screen flow, is pushed from the server to the device, where it's stored. Applications designed using the Mobile Workflow Application are based on the data connections modeled in the MBO tooling, but there is no synchronization used in these applications. Data is either pushed to the device by a DCN, or it's requested by a user in the application; there is no replication that occurs between the CDB and the database on the device. The Mobile Workflow Application does support some offline capabilities:

- An encrypted storage API in the application that can be used to store secure data and/or credentials.
- A queue on the device so that if a user clicks on a submit button in the application but has no connectivity, that submit operation will be queued and then retried when the device connect.

An additional Eclipse plugin, the Mobile Workflow Forms Editor (Figure 3.7), provides developers with a graphical tool for defining the screen flow and for building the UI from a palette of UI elements. Default elements such as list views and detail views can be created in the designer, pre-bound to MBOs. A developer can drag a customer MBO from the navigator to the canvas, and multiple screens will be automatically created: a list view screen of customers, a detail screen to view a single customer, and screens designed for creating a customer or deleting a customer. The developer then formats the screens, chooses which fields are shown, and adds buttons to allow for a query or to submit updates. Before version 2.0, the UI definition was stored as an XML document, and at runtime, the XML rendered it as native UI elements in the device SDK.

The Mobile Workflow Application operates as a container: multiple application packages can be pushed to a single Mobile Workflow Application container. For example, three separate applications—an expense approval, travel request approval, and HR requisition approval application—can all be stored and run within a single Mobile Workflow Application container.

Rebranding Note

In the current Sybase Unwired Platform release, the Mobile Workflow Application has now been renamed to the *Hybrid Web Container*, the Mobile Workflow Forms Editor has been renamed to the *Hybrid App Designer*, and applications designed using the tool are known as *hybrid applications*. However, we'll continue to use the old terminology in this section, as we're covering the history of the product. For a discussion of the current feature set, refer to Section 3.3.

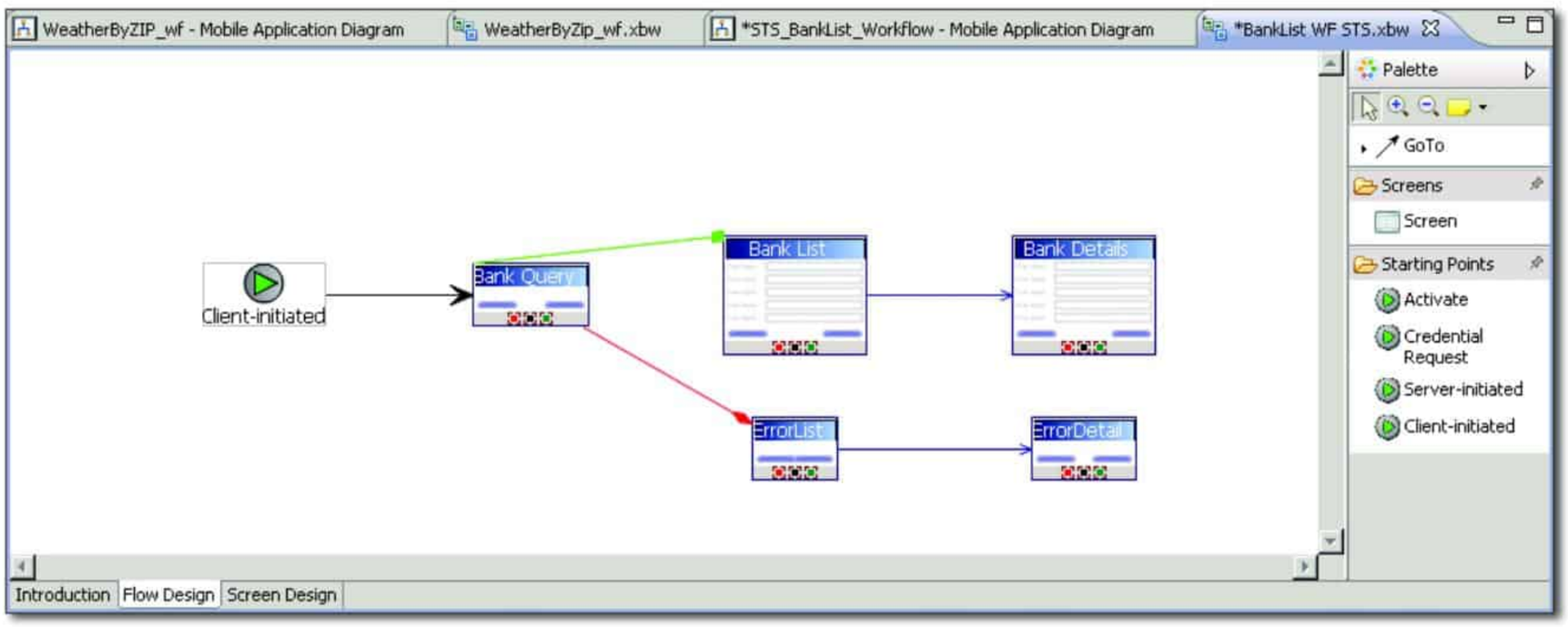


Figure 3.7 The Eclipse-Based Mobile Workflow Forms Editor

There are two modes for applications that are designed using the Mobile Workflow application: client-initiated mode and server-initiated mode. Figure 3.8 shows an example of an application. In client-initiated mode, the end user launches the Mobile Workflow Application, and then selects the specific application package from a list on the home screen. The application reads the XML, builds the UI from the native SDK, and binds the screen attributes to a response message mapped to the screen's MBO. This is essentially the user experience expected when launching a native application from the home screen on the mobile device.

In the server-initiated application example, the screen flow is also designed in the Mobile Workflow Forms Editor, but an additional DCN is configured to look for events with a particular parameter matching value (see Figure 3.9). When the matched parameter is discovered, the server will make a request against the data source for additional information, package the response, and then use a *push* channel to deliver the data package to the device.

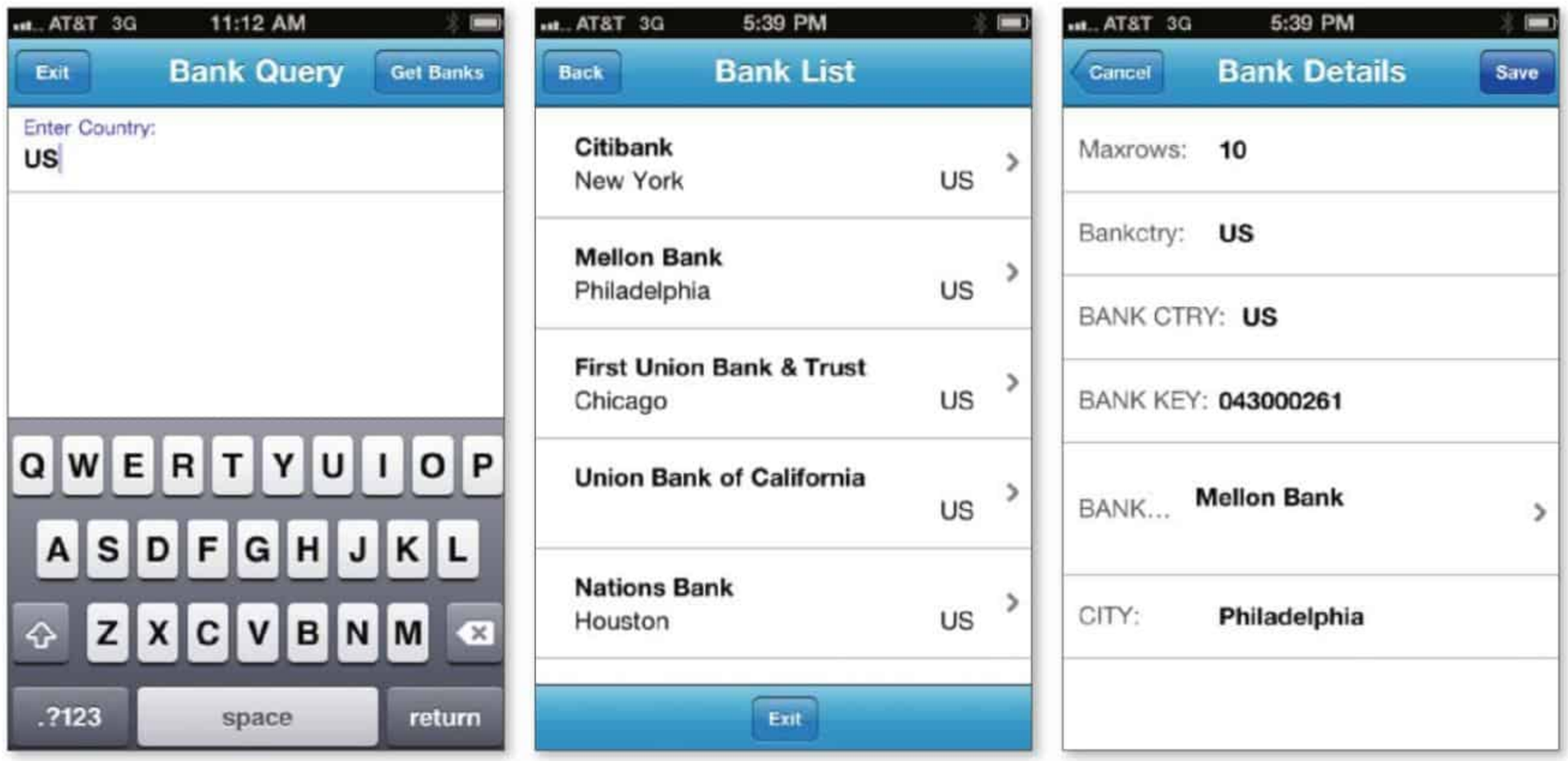


Figure 3.8 Example of a Mobile Workflow Application from Sybase Unwired Platform 1.x

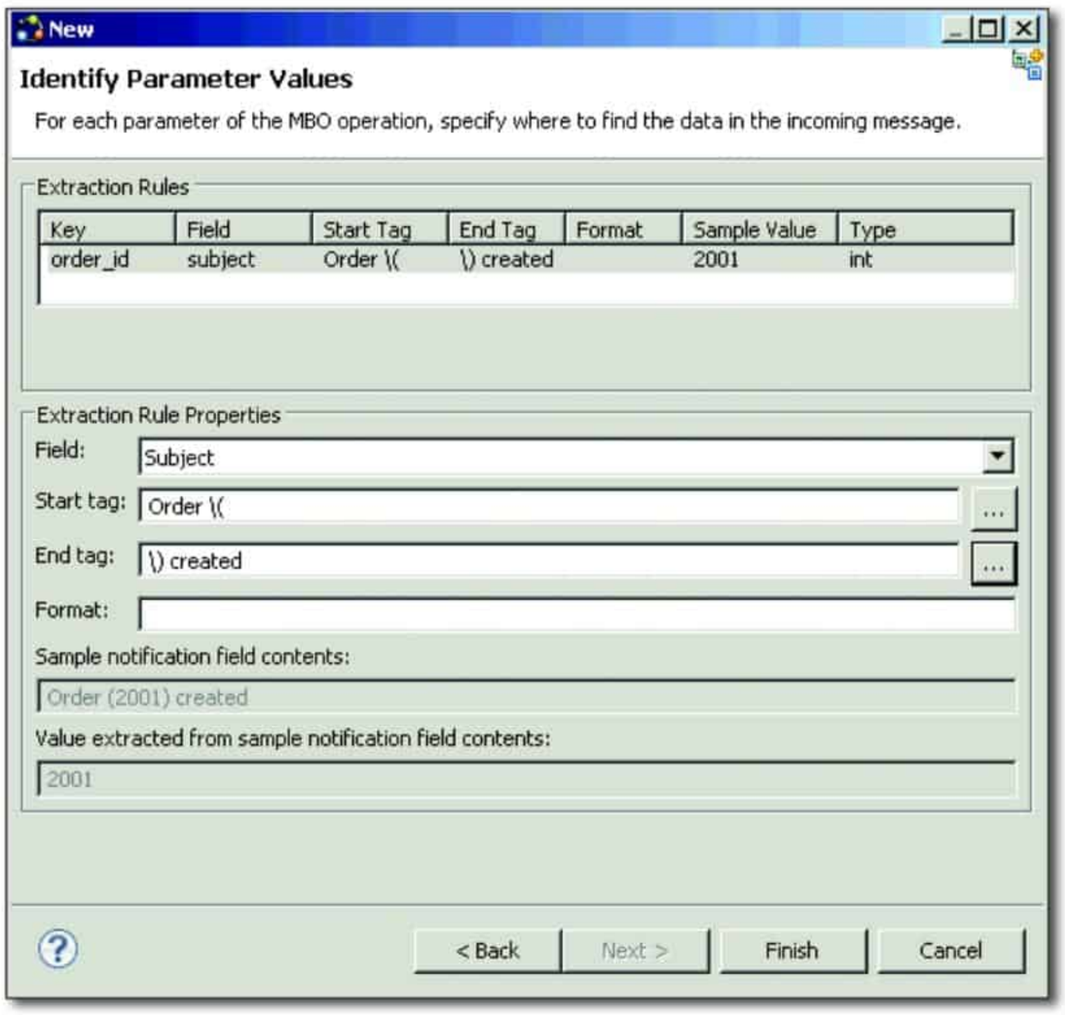


Figure 3.9 Wizard for Configuring Data Change Notification

The Mobile Workflow Application became the core of the Hybrid Web Container in Sybase Unwired Platform 2.0, combining the functionalities of the workflow

APIs (push, secure offline, application versioning) with the cross-platform capabilities of HTML5 and JavaScript, just as HTML5 began to gain widespread traction in the enterprise.

3.3 Sybase Unwired Platform 2.x

SAP issued Sybase Unwired Platform 2.x in Q2 2011, which was the first release to be developed end to end after the acquisition. Sybase Unwired Platform 2.x provided a significant improvement over the previous versions in terms of performance, scalability, and support for typical SAP IT landscape security requirements such as SSO, broad HTTPS support, and network edge authentication. But the defining characteristics of the 2.0.x and 2.1 releases were the addition of the OData SDK and support for the SAP NetWeaver Gateway module, as well as the addition of the Hybrid Web Container to the SDK.

3.3.1 SAP NetWeaver Gateway and OData

SAP NetWeaver Gateway is an SAP NetWeaver add-on module that enables developers to model ABAP content as OData formatted RESTful (representational state transfer) APIs. OData is a Microsoft standard adopted by SAP as the standard for consuming SAP data from multi-channel interfaces: desktop, web, and mobile. OData is used to power consumer web applications, such as Netflix and Facebook, and provides an OData-formatted feed that developers can integrate into their applications.

The Sybase Unwired Platform and SAP NetWeaver Gateway combine to enable the simple consumption of SAP data in mobile applications (SAP NetWeaver Gateway) and enable simple integration of the applications into the enterprise IT landscape (Sybase Unwired Platform).

In the Sybase Unwired Platform 2.1 architecture, Sybase Unwired Platform uses a subset of its typical services to act as a data proxy for service requests from the application to SAP NetWeaver Gateway (Figure 3.10). The application contains a connectivity library for authenticating against the Sybase Unwired Platform proxy, and SSO is handled in the proxy. The communications between the application and the Sybase Unwired Platform proxy are compressed and encrypted, and they use a guaranteed messaging protocol to ensure that messages are delivered regardless of breaks in connectivity. The administrator also registers

("subscribes") to notifications exposed by a SAP NetWeaver Gateway service that can then be pushed to the device. These use a combination of Sybase Unwired Platform functionality and standard operating system (OS) push services (i.e., Apple Push Notification Service [APNS]) to ensure that notifications are delivered from the SAP NetWeaver Gateway to the mobile application.

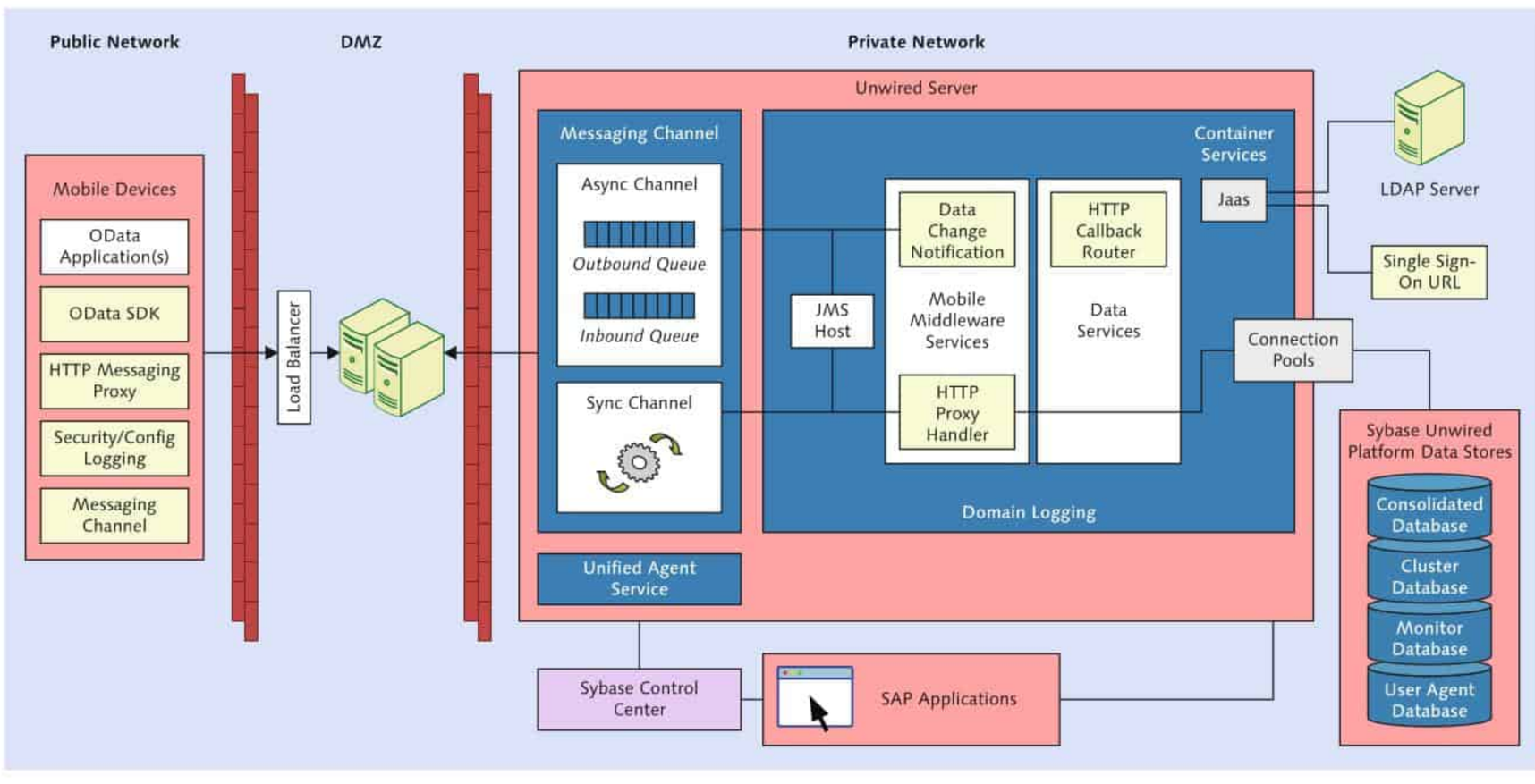


Figure 3.10 Sybase Unwired Platform OData Infrastructure for SAP NetWeaver Gateway

SAP delivered a suite of productivity applications in Q4 2011 based on the Sybase Unwired Platform and the SAP NetWeaver Gateway architecture. These applications use request/response functionality for quick lookups, ad hoc requests, and access to the SAP backend. They do not synchronize data between the Sybase Unwired Platform cache and a database on the device. Therefore, they do require connectivity for transactional events.

Note

For more detailed information about SAP NetWeaver Gateway and OData, see Part III: SAP Mobile Development.

3.3.2 The Hybrid Web Container

The Hybrid Web Container is a native application installed on an iOS, BlackBerry, Windows Mobile, or (as of the 2.0.1 update) Android device. It's based on the

earlier-generation Mobile Workflow Application (discussed in Section 3.2.4) and contains the Sybase Unwired Platform libraries for push, secure storage, SSO, and application versioning. It also contains an embedded HTML5-enabled browser, and instead of rendering the XML documents from Mobile Workflow 1.x, it renders HTML5, JavaScript, and CSS-based (Cascading Style Sheets) applications. This means that instead of the applications being limited in look-and-feel and logic by the XML designer, they are 100% customizable by any web developer. The team dubbed these new applications—written in web languages, but running in the native container—*hybrid applications*. Within a year, they would become the core of many customers' strategy for custom application development (Figure 3.11).

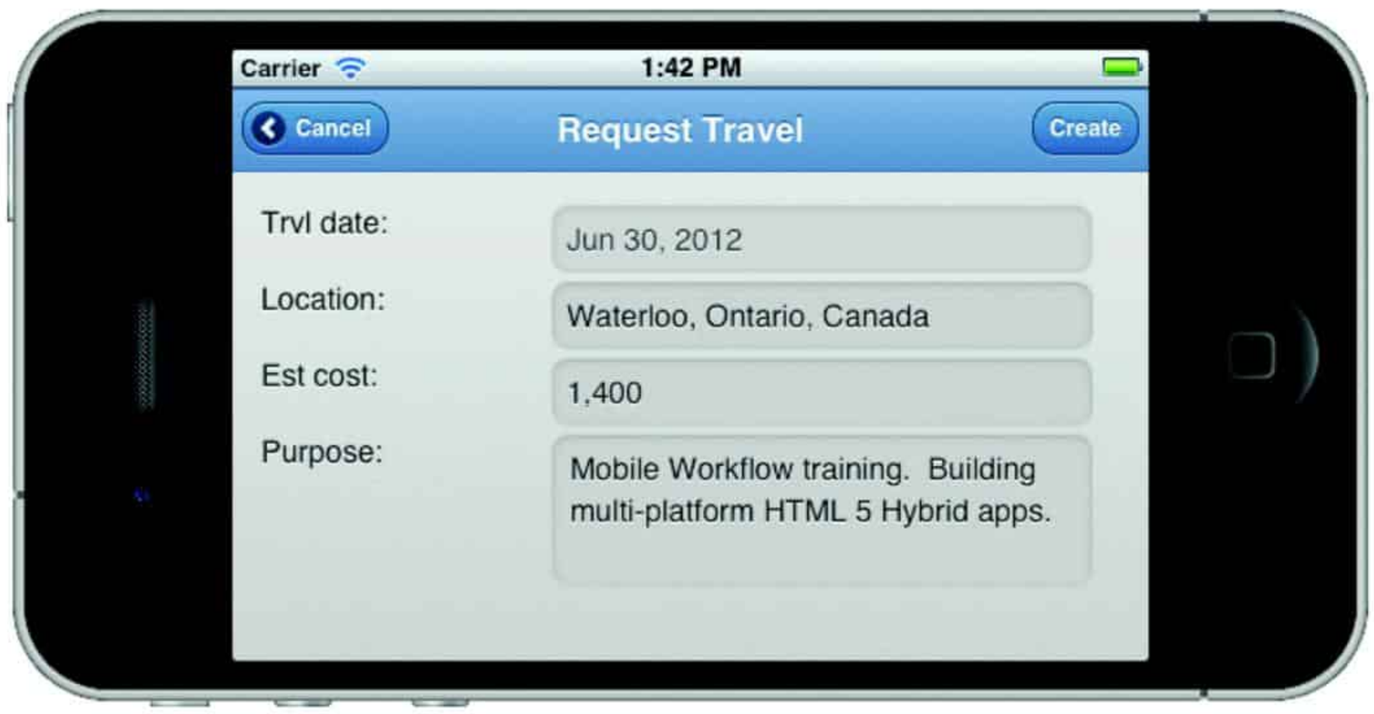


Figure 3.11 Example Hybrid Web Container Application Using default jQueryMobile Look and Feel

Why go hybrid? Between May 2011 and May 2012, the reasons for doing so have not changed, but the market's comparison has evolved from hybrid versus pure HTML5, to hybrid versus native. It's instructive to understand how this happened. The core driver for the hybrid approach is the ability to develop and deploy cross-platform (write-once, deploy-many) applications with mobile platform functionality. An alternative cross-platform strategy could be to use the browser only. When the Hybrid Web Container was first released, the hybrid model was relatively new to the market, and most discussions with customers revolved around when to use a hybrid development approach instead of pure HTML5. HTML5 promised some offline storage capabilities, and while the specification wasn't complete or widely adopted, customers had questions about

whether the Hybrid Web Container was necessary for application development and distribution. As awareness grew about the browser capabilities, the following was found:

- Many mobile applications required native camera functionality, and especially bar code scanning.
- End users expected applications, not bookmarks.
- End users expected push features.
- Out-of-the-box offline capabilities were a key component of an overall mobile applications portfolio.
- Native-grade security is core.

All of these features require some native application functionality not available in the browser on mobile devices. Support for camera features was added to the Hybrid Web Container in version 2.1. To enable additional native device functionality, SAP selected the PhoneGap libraries (discussed in greater detail in Section 3.4.2) to integrate as an open source component; as of version 2.1.3 (June 2012), the Hybrid Web Container ships with linked PhoneGap libraries.

Again, what we saw late in 2011 and through the first half of 2012 was that the conversations changed from comparing hybrid versus HTML5, to hybrid versus native. As customers increased their understanding of what was possible in the Hybrid Web Container, they asked: "Why should I go native at all?" This trend towards cross-platform, hybrid approaches is the most important trend in custom application development of 2012. Given the prevalence of bring your own device (BYOD), and the growth of consumer-facing applications—described by one customer IT manager as "BYOD on steroids"—cross-platform development capabilities are increasingly core to enterprise mobile platform capabilities. The Hybrid Web Container, and also the 3rd party development tools described in Section 3.4, are designed to meet this trend for developing, deploying, and managing employee and consumer facing applications.

3.3.3 Other Key Features

In addition to the support for SAP NetWeaver Gateway, initial release and subsequent enhancements to the Hybrid Web Container, and partnerships with third-party cross-platform tooling providers (discussed in Section 3.4), SAP has invested heavily in integrating Sybase Unwired Platform into the standard SAP

supportability model. SAP maintains standards bodies across the organization chartered with ensuring that its products meet expected criteria around security, performance, usability, lifecycle management, and especially supportability. A few notable additions to the SAP Mobile Platform feature set as a result of these include the following:

- Support for HTTPS across all applications
- Support for CA Siteminder
- Integration with SAP Solution Manager

We'll discuss each of these features in more detail next.

Support for HTTPS Across All Applications

Data communications between mobile applications and the Sybase Unwired Platform server have always been encrypted either by option or requirement, depending on whether the application is replicating data (replication-based synchronization [RBS]) or messaging data (messaging-based synchronization [MBS]). All communications use FIPS-140-compliant libraries (Federal Information Processing System). Prior to Q2 2012, MBS data was encrypted end to end from the server to the device, which was suitable from a security perspective but didn't enable customers to use their own network-edge authentication mechanisms, such as CA Siteminder, to inspect the traffic as it passed through the firewall as they could otherwise if the traffic were using HTTPS (RBS applications could use HTTPS). Therefore, while the customer could trust SAP's security protocol, there was no mechanism at the network edge for auditing and limited ability to protect against denial-of-service attacks. The MBS protocol was updated in Q2 2012 to support HTTPS for applications on all platform OSs. (iOS-based applications had been supported by year-end 2011.)

Support for CA Siteminder

CA Siteminder is a common security agent installed at the customer's network edge to validate traffic entering the network. The Siteminder agents have typically been installed to protect web traffic, and extending the rules defined on the Siteminder policy server inside the firewall to apply to traffic coming from mobile applications is a natural fit. SAP customers also commonly use the Siteminder token generated for the validated session to create a single sign-on (SSO) token that can be used against the backend SAP system.

SAP partnered with CA Technologies in Q2 2012 to validate an integrated architecture (Figure 3.12) in which credentials passed from the device are appended to headers on the request to the server, captured and validated by the Siteminder web agent at the network edge, and then used to generate a session token managed by Sybase Unwired Platform to generate and reuse the SSO token for connecting to SAP.

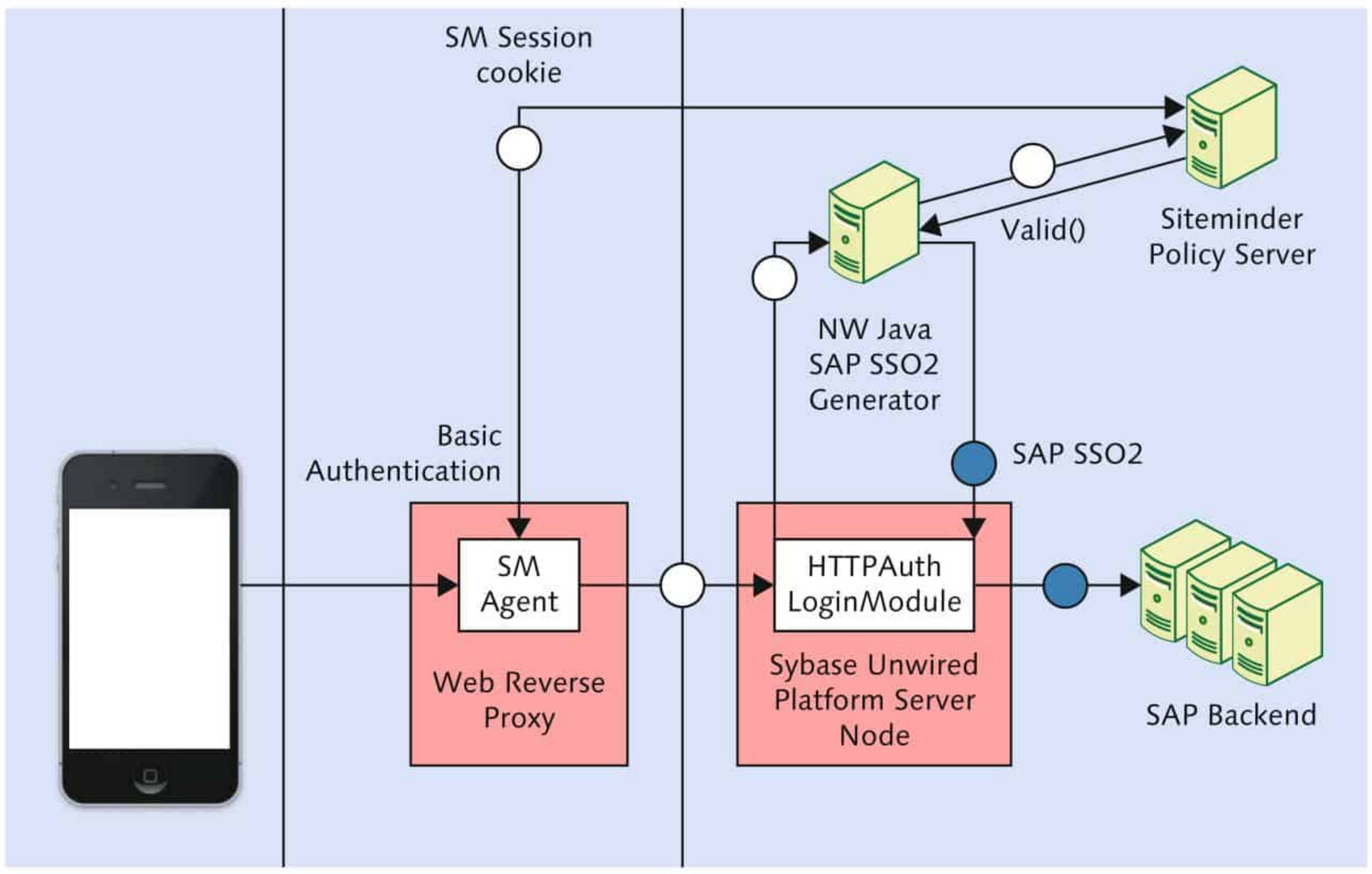


Figure 3.12 Reference Architecture for Sybase Unwired Platform and CA Siteminder-Protected Services

Integration with SAP Solution Manager

SAP Solution Manager is a tool delivered by SAP to enable the manageability of software on the SAP landscape. Its feature set includes version management, transport, diagnostic tracing and root-cause analysis, and auditing, and it's the primary interface for SAP's Active Global Support (AGS) organization to provide technical support. Support for SAP Solution Manager features around license auditing were provided in Q2 2012, and additional capabilities for end-to-end trace and CTS+ (enhanced Change and Transport System) integration were planned for the second half of 2012 at the time of this writing.

3.4 Integrations with Third-Party Cross-Platform Tools

In Q1 2012, SAP announced three partnerships in a single event around support for cross-platform development tools. The three key vendors involved were Adobe, Appcelerator, and Sencha. These vendors deliver solutions that enable the development of cross-platform applications and have strong developer communities in the consumer mobile application space and increasingly in the enterprise mobile application space In a number of instances, SAP Mobile Platform customers and partners were already finding ways to integrate their platform-based code and applications with these tools, so the announcement was significant both in its validation of these projects and in signaling SAP's commitment to engaging broader developer communities outside its existing ABAP core.

Cross-platform tools refers to tools and/or runtimes that enable the development of a single client application, or code line, that can run on multiple device types. A number of approaches to accomplishing this have been pursued by vendors between 2010 and 2012, but by 2012, JavaScript has been broadly adopted as the de facto language for cross-platform application development, and vendors that had previously pursued alternate tracks are shifting their focus from previous investments to try to join the majority.

When we talk about developing mobile applications, the most common use of JavaScript is to power the UI framework. When developers write native applications for iOS or Android, they use the native SDK to build the UI. Each of these platforms provides a set of standard UI building blocks—labels, tables, sliders, and switches—and a set of navigational controls that make up the usual screen-flow of the application—screen, detail view, navigate forward, go back, and so on—and the animations associated. The UI elements and navigational controls are combined to make up the UI; for example, the developer adds a SUBMIT button to the upper-right corner of the screen, and once clicked, the application removes the current screen and animates a transition back to the previous list. This combination of elements and navigation makes up a UI framework.

3.4.1 Sencha Touch and jQueryMobile

Now think about how a web page works that is developed in HTML5 and JavaScript. There are standard UI elements, such as `<div>`, `<table>`, and `<row>`, and standard navigational actions, such as clicking on a link to present a new

resource. But a web page navigation doesn't behave like most mobile applications' navigation because there is no concept of a screen or moving from screen to screen. To fill this gap, a number of open-source projects sprung up in 2009 and 2010 to create a JavaScript-based UI framework targeting mobile devices. The most popular are Sencha Touch and jQueryMobile. SAP UI5 is also relevant in the SAP ecosystem. All projects provide a web language experience for developing mobile applications.

Sencha Touch and jQueryMobile differ in the way they provide a web language. Sencha Touch is a pure JavaScript (100% code) means of creating UI elements, model objects, and navigating between screens. Its basic application uses only two HTML5 tags to run in the browser. To see a sample Sencha Touch 2.0 application, we recommend the Sencha Touch 2.0 *Building Your First App* guide.[2]

jQueryMobile, on the other hand, uses HTML5 tags to create UI elements during the development phases. Each `<div>` is assigned a data role from the framework; that is, a page, header, footer, content, button, list view, and so on. Each data-role type can be modified by a set of configurable attributes. To see a sample jQueryMobile two-screen application, we recommend the jQueryMobile *Anatomy of a Page* guide.[3] Both approaches are supported for applications running in the Hybrid Web Container. Sybase Unwired Platform ships with the most recent jQueryMobile library in the SDK as an open-source component, and the Hybrid App Designer (the HTML5-enabled design tool based on the earlier Mobile Workflow Forms Editor) outputs jQueryMobile-based applications when it generates code. (SAP published a whitepaper and code samples in Q2 to the SAP Community Network (SCN) demonstrating how to use the two technologies together.) As of mid-2012, Sencha is very popular for high-fidelity look-and-feel applications due to the improved navigational transitions accomplished by the 100% code approach. jQueryMobile provides a very simple configurable approach for applications with list view—detail screen patterns—and it relies more heavily on the combination of HTML5, JavaScript, and CSS for the application's look-and-feel.

3.4.2 PhoneGap (Apache Cordova)

PhoneGap (now Apache Cordova) is an open-source project created by Nitobi, which was acquired in Q4 2011 by Adobe. The PhoneGap code line was donated

2 *http://docs.sencha.com/touch/2-0/#!/guide/first_app*

3 *http://jquerymobile.com/demos/1.1.0/docs/pages/page-anatomy.html*

on the date of the acquisition to the Apache Foundation, where it was renamed Callback, and then Cordova. It's licensed under version 2.0 of the Apache license. SAP announced a partnership with Adobe in Q1 2012 around extending reach on the SAP Mobile Platform to PhoneGap developers, and in Q2 shipped the Apache Cordova libraries bundled in the Hybrid Web Container.

Brian Leroux of Adobe (formerly Nitobi) blogged that the PhoneGap project had two stated goals: 1) the web as a first-class development platform, and 2) the ultimate purpose of PhoneGap is to cease to exist.[4] In other words, the open-source PhoneGap project was intended to fill a gap in the browser capabilities on mobile devices that would someday be supported with the HTML specification. (Perhaps this will become the theme of HTML6.) In practice, PhoneGap enables JavaScript developers to connect to device APIs (i.e., GPS, Accelerometer, file system, etc.) in the native SDK on each OS from their web-language applications running in a native container application. The developers also have the ability to extend the PhoneGap framework to add custom binding between additional native code and JavaScript. For example, they can add a third-party bar code scanning library to their container application and bind the function calls to their own JavaScript.

The PhoneGap container is similar in construction to the Hybrid Web Container, but there is really only one overlapping part: the web view. The PhoneGap container is comprised of three essential parts:

- Native PhoneGap library (provides binding to native device SDK)
- JavaScript PhoneGap library (provides JavaScript function calls)
- WebKit-based webview

The Hybrid Web Container has complementary parts at these layers:

- Native Sybase Unwired Platform libraries (provides connectivity, security, authentication, storage, application versioning, and push)
- JavaScript API library (provides JavaScript function calls to data from the server, local storage, etc.)
- WebKit-based web view

---

4 *http://phonegap.com/2012/05/09/phonegap-beliefs-goals-and-philosophy/*

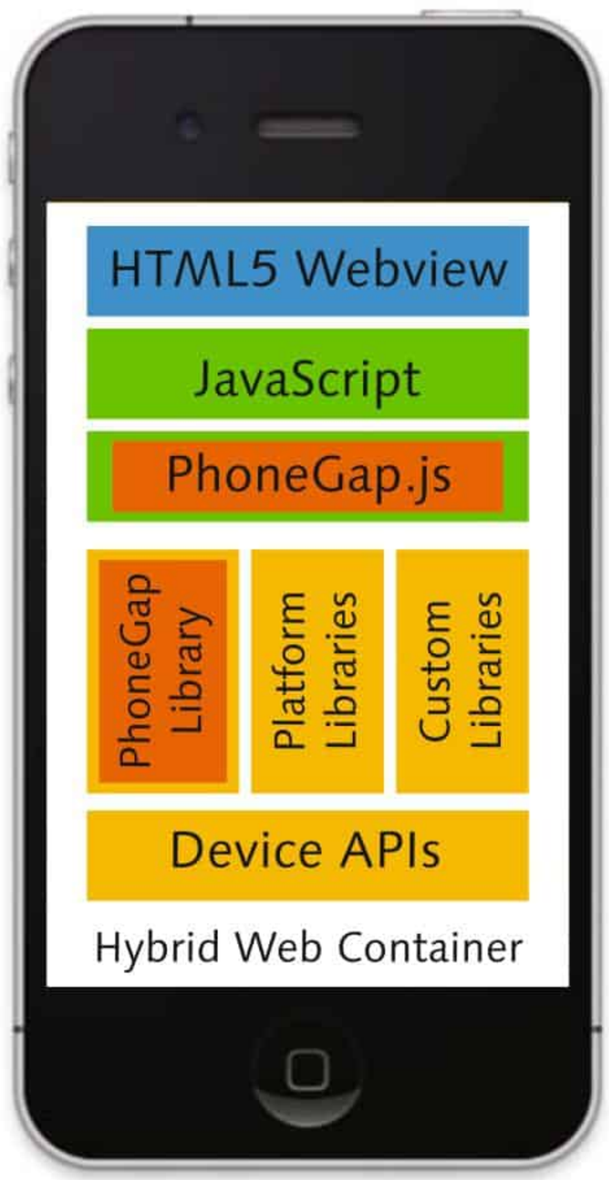


Figure 3.13 Hybrid Web Container Components, with PhoneGap (Apache Cordova) Embedded

The bundled PhoneGap plus the Hybrid Web Container provides the following developer feature set (Figure 3.13):

- Native PhoneGap library
- JavaScript PhoneGap library
- Native Sybase Unwired Platform libraries
- Sybase Unwired Platform JavaScript API
- WebKit-based webview

The combined set of libraries enables developers to build mobile applications with web languages, calling industry-standard APIs for device features, fully integrated into the enterprise platform services. Further, it provides a path for custom-developed solutions that map SAP Mobile Platform features currently provided only in native libraries to JavaScript-based applications. While in mid-2012, the limitations on hybrid application capabilities are essentially related to

data volumes and the ability to plug into the SAP Mobile Platform offline-sync feature set, the combined Hybrid Web Container and PhoneGap solution has been used in custom projects to fill this gap.

3.4.3 Appcelerator

In the same Q1 Partnership announcement, SAP announced that it would be working with Appcelerator to extend the SAP Mobile Platform into the Appcelerator Titanium development framework. Appcelerator takes a third approach to cross-platform applications with its Titanium solution, but it's also JavaScript-based to the developer. The developer writes the application in JavaScript (similar to Sencha Touch) but then compiles the code in the tool to generate a native application for iOS and Android code lines. The benefits are obvious: one single code line, native look-and-feel, and UI performance.

Appcelerator accomplishes this in a manner similar to PhoneGap; it provides proxy classes that map native code to the JavaScript layer. Essentially, Titanium subclasses are the classes in the native SDK (e.g., `UITableViewController`), and though the developer doesn't need to know the respective iOS or Android classes when creating a table view, they will be automatically created and used by the native application. Custom business-logic code (not related to UIs) may also be written in the JavaScript. To extend the Titanium framework, the developer adds his own native code to the project and then creates a proxy class to complete the mapping.

A few Appcelerator plus Sybase Unwired Platform projects had grown up organically before the acquisition, including a 900+ iPad deployment of offline-enabled mobile sales applications. The application enables sales representatives to visit luxury retail boutiques, display rich media marketing materials that are distributed in bulk based on location and brand, search for products, and order based on size and style. The process includes signature capture, and the whole transaction can be completed and saved offline—for example, if the store is in a mall with poor or no reception—and then synched to the order system when connectivity is restored. The SAP Partner involved in the project developed a tool for generating the Titanium proxy classes for the MBOs in the application, simplifying the development process.

Appcelerator also provides a reference application for binding the OData-formatted content exposed by SAP NetWeaver Gateway in the Titanium framework.

3.5 Introduction to SAP Afaria

Today's mobile landscape is increasingly complex due to large numbers of device types, applications, OSs, and ever-growing security threats. SAP Afaria is a solution to manage and secure large-scale deployments of mobile devices.

As an MDM (mobile device management) solution, SAP Afaria can help in three critical functions in a mobile deployment:

- Device management
- Security management
- Application management

SAP Afaria is a flexible solution that offers a host of deployment options. You can get it as a hosted service, or you can deploy it in-house. In in-house deployment scenarios, SAP Afaria can be configured to work both behind the firewall and in your DMZ. All of SAP Afaria's functions can be accessed from the administrative console. The administrative console is an application that runs in the browser of your choice offering an intuitive interface and streamlined workflows for everything from device provisioning, assignment, and scheduling to policy creation and management. New reporting and analytics allow you to track the number of devices by OS, carrier, and manufacturer. This results in a better understanding of roaming activity, popular applications, and compliance issues, plus a simplified device management experience.

We discuss the functionality and use of SAP Afaria in more detail in Chapter 4. In this section, we offer a high-level introduction to the previously mentioned critical focuses of SAP Afaria: device management, security management, and application management.

3.5.1 Device Management

SAP Afaria helps you remotely mange all of the devices in your landscape from one central location. SAP Afaria ensures that mobile devices are configured properly and lets you securely and centrally define and maintain system attributes preferences and settings for remote devices. This includes backup and restore of the device, providing connection settings, disabling certain features of a device (such as the camera feature), setting up the synchronization option for your enterprise groupware (contacts, calendar, mail), and so on. SAP Afaria can work with

the certificate authority (CA) of your choice to provide certificates for devices and applications.

3.5.2 Security Management

SAP Afaria can help enforce your corporate IT policies with regards to security for both BOYD and corporate-owned devices. It can centrally enforce power on passwords with IT-defined strength, set the password expiry time, set up lock times, and set lockdown after a predefined number of failed attempts. SAP Afaria can also encrypt data that resides on devices. If a device is lost or stolen, SAP Afaria can wipe the device remotely.

3.5.3 Application Management

SAP Afaria makes it easy to deploy and manage enterprise applications across your mobile workforce. It delivers in-house applications (by hosting an internal application portal) over the air with distribution control and reliable delivery. It lets you identify applications for mandatory download based on each user's role and responsibilities, while allowing downloads of suggested applications. SAP Afaria also helps in configuring default settings for SAP mobile applications. SAP Afaria sets Sybase Unwired Platform settings such as host, port, and login method so that end users don't have to type this information manually (a task especially difficult on a smartphone). This results in faster adoption and the need for less IT support.

3.6 Mobile Services (mCommerce)

Mobile Commerce is a rapidly growing services market in which Sybase has been a leader before and after the SAP acquisition. Pre-acquisition, Sybase operated a business unit entitled *Sybase 365*, which focused on a variety of business related to messaging and telecommunications carrier services. This included short message service (SMS) and multimedia messaging service (MMS) inter-operator hubs, connecting messages across a network of nearly 1,000 carriers, and reaching more than 5 billion mobile subscribers.

An offshoot of the network inter-operator was the application-to-person (A2P) messaging solution. This technology enables any web server application to interact with a phone user via SMS, just as if they were another phone user. The opportunities for this type of phone-to-application interaction are massive. Sybase 365's customers included Facebook, for which it carried SMS-based news feed updates (and pokes), and Nokia, which carried out its marketing campaigns via SMS. MMS can also be used for content delivery, and carriers used Sybase 365 to deliver wallpapers and ringtones to end users on their network.

In order to use SMS to complete transactions such as content purchases, a server-side solution is required to maintain the session, because the SMS reader on a user's phone has no "state." This means that it has no memory of the previous message, even if it's still visible—it can't read the messages and make sense of them in a multi-step process. There's nowhere on the phone where you can write logic that says "if my dentist sends an appointment reminder by SMS, check my calendar and reply y/n if the time slot is reserved." For there to be any meaningful transaction, there will need to be two-way communication, multiple messages, and so there needs to be an application running on the server that can start a session and translate the SMS to connections to the backend.

For example:

1. The user enters "ACT" to start a get-balance session.
2. The server starts the session and sends a password challenge.
3. The user sends the password.
4. The server checks the password, makes a request to the backend for the current balance, and then sends the current balance.
5. The user enters "TR" for recent transactions.
6. The server makes a request to the backend for the five most-recent transactions, and then sends the information to the device.

Enabling this sort of multi-step transactional interaction between the user and phone is the core of the Sybase (now SAP) mCommerce solutions. The Mobiliser mCommerce framework was designed to enable financial transactions from mobile devices at a large scale: millions of end users per system. In developed markets, the prevalence of payment systems, ATMs, and credit cards have made mobile payments a convenience to the end user; in the emerging markets, the

phone is the primary electronic interface to the connected world. Telecommunications carriers and banks have partnered with Sybase to manage financial transactions executed from devices on their network. The Mobiliser solution includes a transactional database that acts as a system of record, reducing the load on the backend system; the bulk updates are periodically batched to the backend for clearing.

The Mobiliser mCommerce solution has been used for mobile banking (consumer and commercial), payments (in-network top-ups, as well as vendor-network payments), and money transfers (person-to-person payments, and especially remittances).

Note

mCommerce applications are not a primary focus of this book, but, as Mobiliser is an important part of the SAP Mobile Platform, it's worth mentioning in this overview chapter.

3.7 Summary

This chapter provided an overview of the SAP Mobile Platform, focusing on its history and main elements. SAP's strategy is to provide a single, integrated mobile platform technology for its ecosystem's mobile application development, deployment, and management.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W