

Mobile Development Tools

Microsoft Dynamics™ Mobile
Development Tools
White Paper

Extending Business Solutions to the Mobile Workforce

Abstract

This white paper gives an overview of the Mobile Development Tools, describing the architecture and how the tools can be used to develop and deliver mobile applications for business solutions.

Version: June, 2007

Microsoft

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Sample code contained in this white paper is made available as is, without warranty, either express or implied. The entire risk of the use or the result from the use of this code remains with the user.

© 2007 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, ActiveSync, MapPoint, Microsoft Dynamics, Outlook, SQL Server, Visual C#, Visual Studio, Windows, Windows Mobile, and Windows Server are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Table of Contents

Enabling the Mobile Workforce	2
Microsoft Dynamics Mobile Development Tools	2
Mobile Framework	2
Developing Mobile Applications with the Mobile Framework	4
Behind the Mobile Framework.....	4
Developing Tasklets.....	5
Orchestrating Tasklets.....	5
Orchestration Example.....	5
Scaling Applications with Orchestration.....	7
Talking to the Server	8
Architecture Overview	9
The Client.....	9
The Middle Tier.....	9
The Server.....	10
Data Transfer to the Client.....	10
Getting Data from the Business Solution to the Client	10
Sending Data from the Mobile Device to the Business Solution.....	11
Data Transfer Back to the Server	11
Dispatching Request Documents and Controlling Costs	12
Conclusion.....	12

Enabling the Mobile Workforce

With mobile devices becoming increasingly prevalent in today's business world, companies are realizing the benefits of giving the work force access to their business solution through these devices— when and where they need it. Making the business solution services and information available from a mobile device can improve productivity and lower costs. The traveling salesperson, the field service engineer, and the warehouse worker all benefit from being connected to their company's ERP infrastructure:

- There is no need to fill out forms and hand them into the back office, because information is registered at the source.
- There is no more out-of-date data in the ERP system, because information is sent as soon as a connection is available.

However, just providing access to your business solution is not enough. ERP systems can be quite large, containing a vast amount of information and functionality. Most field workers have focused, specific tasks, and they do not need access to the entire business solution. In fact, providing them with access to everything can make their tasks more confusing and time consuming. In addition, recreating all business solution functionality on a mobile device could overload the device. Ideally, you want to only provide workers with the specific information and functionality that they need.

Another problem in the mobile environment is network connectivity. There are times when a network connection is not available, especially if the mobile worker is moving in and out of connectivity zones. To do their job effectively, these workers need access to the business solution functionality, even when they don't have a network connection.

The challenge is to develop a mobile platform that:

- Makes it easy to build role-based and task-driven applications.
- Makes it easy to build simple and intuitive user experiences.
- Is designed for the limited form factor of a mobile device.
- Works in an occasionally connected fashion.
- Is flexible and configurable to allow for customer and partner customizations.
- Has seamless server integration.

Microsoft Dynamics Mobile Development Tools

The Microsoft Dynamics Mobile Development Tools consist of two main parts:

- The Mobile Framework for developing mobile applications and,
- The Mobile Server for enabling communication between the mobile application and the ERP system.

The following sections describe the Mobile Framework and how you can use it to develop your mobile applications and the Mobile Server and how you can use it to connect your application to a Microsoft Dynamics AX business solution.

Mobile Framework

The Mobile Framework is an environment for developing or customizing mobile business applications for Windows Mobile 5.0 and 6.0 devices that connect to business solutions, such as Microsoft Dynamics AX. The Mobile Framework extends the .NET Compact Framework and consists of:

- A set of managed APIs that enable developers to create scalable mobile applications using C# and XML.
- A sample mobile application.

-
- A tool for localizing strings in the mobile application.
 - A tool for compiling application definitions into a managed assembly.
 - A set of standard building blocks (called tasklets), which are required by most applications.

Developing Mobile Applications with the Mobile Framework

The Mobile Framework enables you to build modularized mobile business applications tailored to specific roles in your company. Using the Mobile Framework (an extension of Visual Studio 2005), developers can quickly create mobile applications from reusable components called tasklets. Tasklets can be combined in different ways to cater to the needs of individuals. This RoleTailored architecture leverages the power of Windows Mobile devices and helps mobile users become more effective and efficient in their daily tasks.

Using a combination of C# and XML, you can build customized, role-based mobile applications with a look and feel consistent with other applications on the device. The mobile applications consist of a series of intuitive, menu-driven screens that a mobile user navigates to view information and exchange data with the business solution.

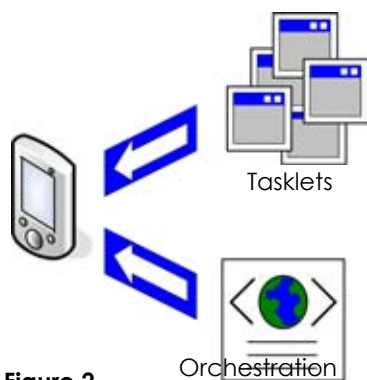


Figure 2

Behind the Mobile Framework

The work performed on a mobile device is usually very task-oriented and specific to the user's job or role. A salesperson places orders, a service technician registers time and materials, and a warehouse worker picks orders. The mobile device should be tailored to the role of the individual user or group of users.

To create customized mobile applications the conventional way, you need to develop a separate Windows application for each role-based scenario and then load these applications on the mobile device. With the Mobile Framework, you can divide a business solution into incremental parts—developing each part as

a separate .NET type. These parts are called **tasklets**. Using XML and a concept called **orchestration**, you can piece these tasklets together into different configurations. This enables you to create personalized, role-based applications that you can then install on a device-by-device or role-by-role basis.

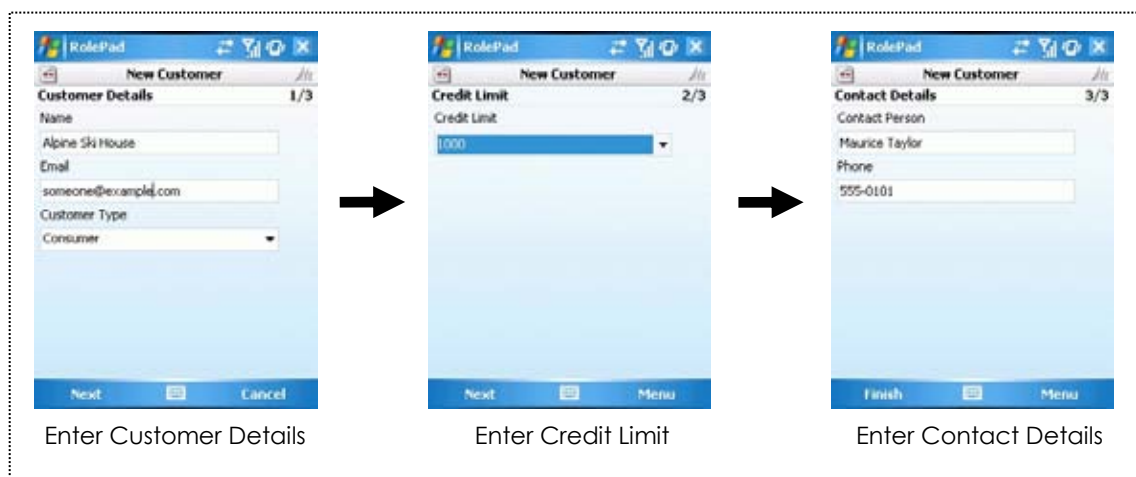


Figure 3: Example Orchestration of Tasklets for Creating a Customer

Developing Tasklets

Tasklets are created in Visual Studio using the Mobile Framework. A tasklet is a .NET type, which most likely lives in its own assembly. Each tasklet performs a specific action, which typically represents part of a larger task performed by the mobile user. For example, a tasklet could display a list of customers, or details about a specific customer. Consider the steps that are involved in placing an order. This action typically involves choosing the items to order, entering shipping information, and finally sending the order. When developing an application, you can develop each of these steps as a separate tasklet.

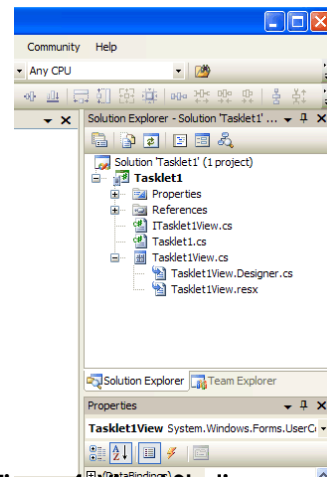


Figure 4: Visual Studio

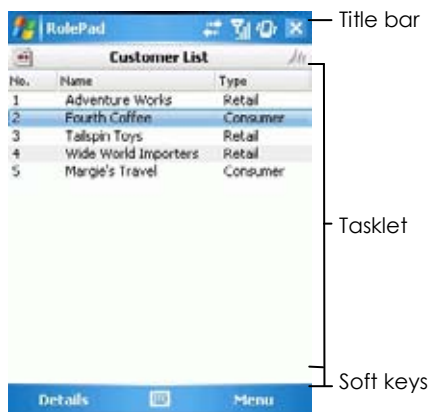


Figure 5: Customer List UI

Data Handling

Tasklets can be programmed to perform various actions for handling data. A tasklet can display data from a Microsoft SQL Server Compact Edition database on the device, accept input from another tasklet, and output data to a request document.

Tasklet User Interface

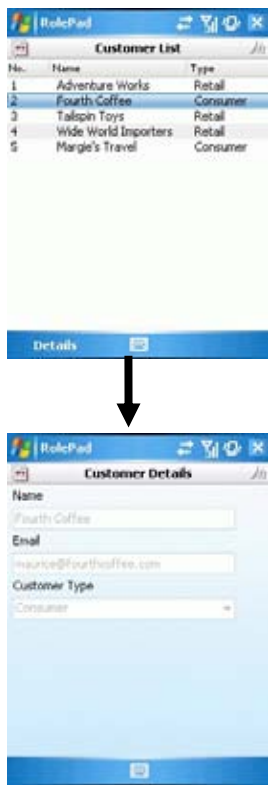
All tasklets use a common user interface (UI) to display information on the mobile device. The Mobile Framework uses a template that provides you with the standard Windows mobile layout, including the title bar and soft key menu. As a developer, you only have to be concerned with the content and layout specific to the tasklet.

Orchestrating Tasklets

Orchestration controls the flow of and interaction between tasklets, determining how the mobile user navigates through the application. The orchestration is developed in XML and is stored in the UserRole.xml file. Because the input to and output from tasklets is managed in the orchestration, no additional tasklet coding is required to use them in different mobile applications. This enables you to reuse tasklets in other flows within the same application or even across applications. Individual tasklets can even store a set of instance configuration data in UserRole.xml file, so that the actual behavior of the tasklet can change depending on the setup.

Orchestration Example

To understand orchestration, look at the following example of a simple mobile application shown in Figure 6. This application consists of two tasklets. The first tasklet, called "Customer List," displays a list of customers from a database. By selecting a customer in the list and tapping **Details**, you can view information about that customer in another tasklet called "Customer Details". The UserRole.xml file that defines the orchestration for this application is shown to the right.



```

<?xml version="1.0" encoding="utf-8" ?>
<userRole text="RolePad">
  <orchestrations>
    <orchestration text="RolePad" name="Sample App"> ← 1
      <tasklets>
        <tasklet name="CustomerListTasklet" ← 2
          type="CustomerListTasklet.CustomerListTasklet,
            CustomerListTasklet">

          <outputMappings>
            <outputMapping propertyName="SelectedCustomer" ← 3
              stateKey="CustomerID"/>
          </outputMappings>

          <actions>
            <open text="Details" name="CustomerDetail" ← 4
              tasklet="CustomerDetailTasklet">

            <inputMappings>
              <inputMapping propertyName="CustomerID" ← 5
                required="true" stateKey="CustomerID"/>
            </inputMappings>

            </open>
          </actions>
        </tasklet>

        <tasklet name="CustomerDetailTasklet" ← 2
          type="CustomerDetailTasklet.CustomerDetailTasklet,
            CustomerDetailTasklet">
        </tasklet>
      </tasklets>
    </orchestration>
  </orchestrations>
</userRole>

```

Figure 6: Example Tasklets and UserRole.xml file

The following describes some of the lines in the UserRole.xml file (numbered in Figure 6), to help illustrate how this works:

- The `<orchestration>` element (line 1) defines the orchestration. An orchestration specifies the relationship between one or more tasklets. In this example, there is only one orchestration, but in a typical mobile application there are a number of orchestrations. For example, you might have an orchestration for adding a new customer or placing orders. Each of these would be defined in its own orchestration.
 - The `<tasklet name="" type="">` (line 2) declares a tasklet based on its fully qualified type name. Each tasklet must be declared in the orchestration. The first tasklet listed in the first `<orchestration>` element of the UserRole file is the tasklet that opens when the application starts. So in this example, the Customer List tasklet is placed first.
-
- Note:** A tasklet can be used in more than one orchestration within a UserRole.xml file. The name attribute lets you reference the declaration from other locations in the orchestration, so you only have to declare it once.
-
- The Customer List tasklet outputs an ID to identify the selected customer (line 3). The `<outputMapping>` element is used to transfer this ID to the Customer Details tasklet.

- The `<actions>` element (line 4) specifies actions a user can take from a tasklet screen. This determines the soft keys and menu items and to which tasklet they link. In this example, to link from the Customer List tasklet to the Customer Details tasklet from a **Details** soft key, the following is added `<open text="Details" name="CustomerDetail" tasklet="CustomerDetailTasklet">`, where `text="Details"` specifies the soft key text and `tasklet="CustomerDetailTasklet"` calls the CustomerDetails tasklet declared earlier.
- To display data from a specific customer in the Customer Details tasklet, an `<input>` element (line 5) is added to map the customer ID output from the Customer List tasklet to the Customer Details tasklet.

Scaling Applications with Orchestration

The advantage of orchestration and the `UserRole.xml` file is that once you have your tasklets, you can extend or modify an application using XML, without having to modify any code. This takes the responsibility of scaling an application out of the developer's hands and gives it to a consultant. C# programming experience is not required.

To illustrate this, the example application in Figure 6 has been modified to include a tasklet to view a contact person for a selected customer from the Customer List. In this example, the tasklet is called `ContactTasklet`. To make this modification in the application, you change the orchestration to include a line that declares the `Contact` tasklet and an action to open the tasklet from a menu item called `Contact`.

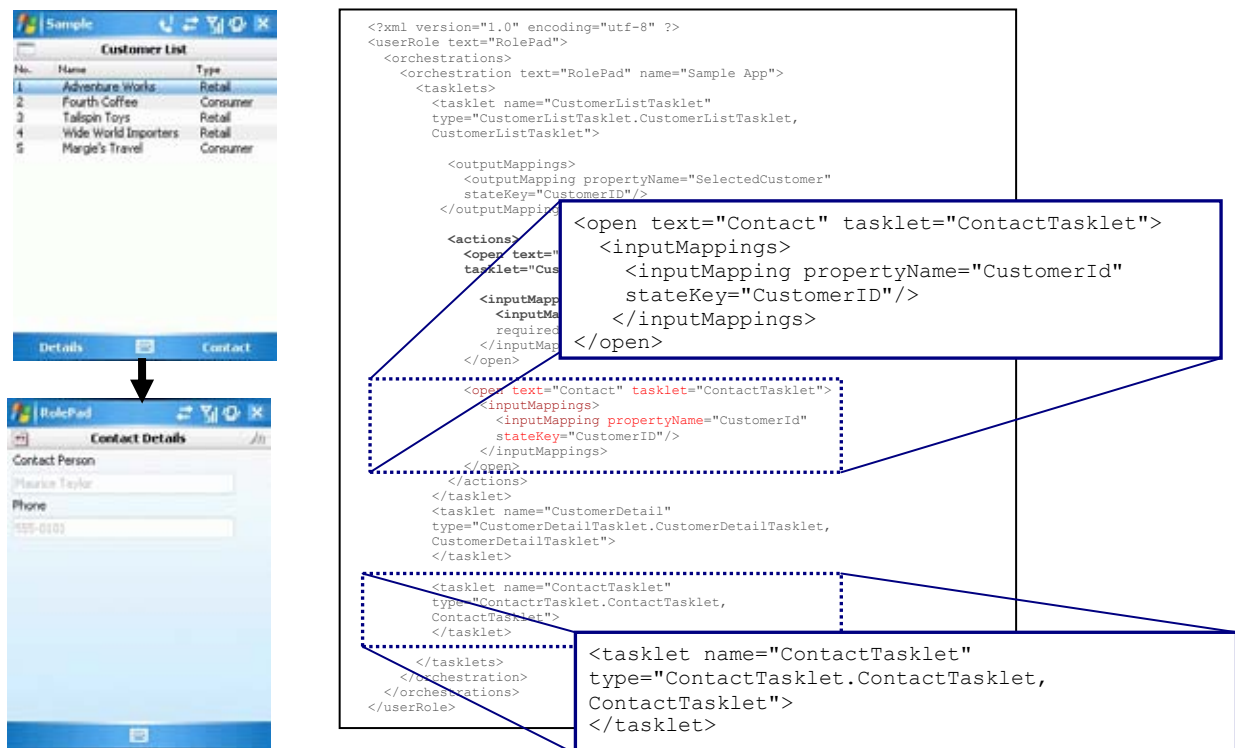


Figure 7: Adding a New Tasklet to an Orchestration

This way of building applications enables reuse of components and orchestrations, so that multiple role specific solutions do not become a nightmare to maintain. It also clearly separates the development and process work, so that people close to the process can assemble the application. Figure 8 gives an example of a simple RoleTailored application built using this methodology.

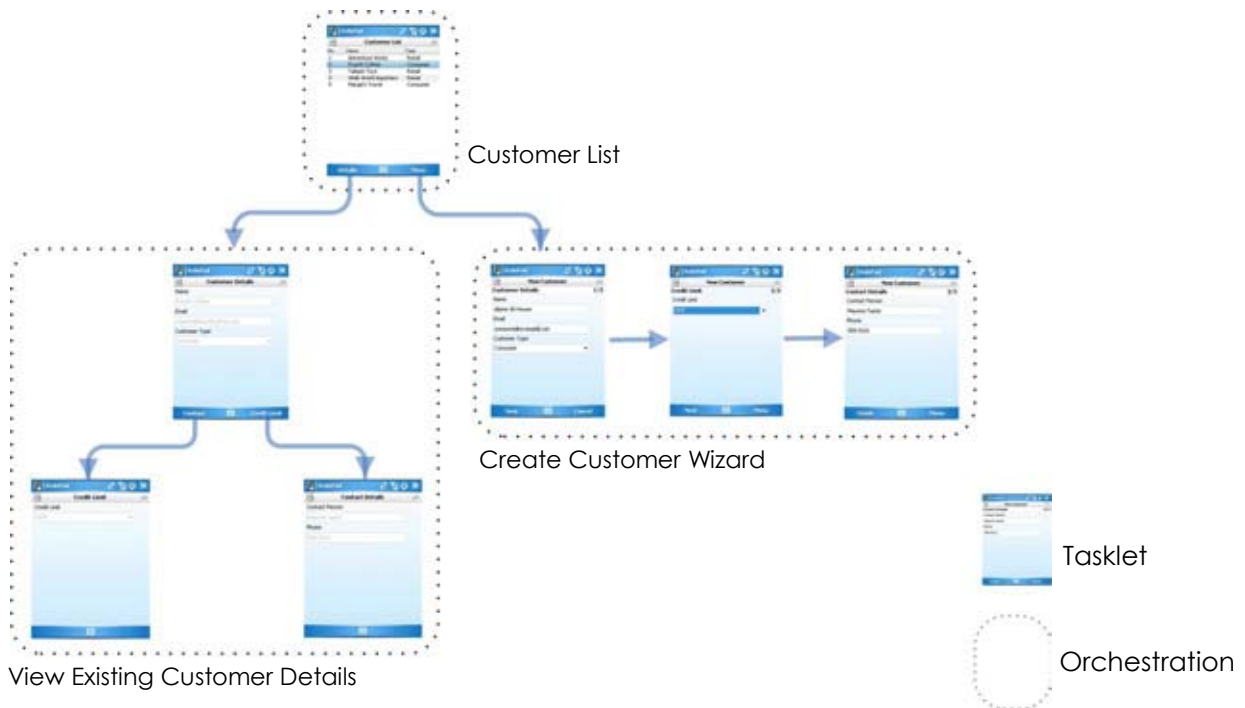


Figure 8: Adding Orchestrations to Build an Application

Talking to the Server

The Mobile Development Tools also provide the overall architecture for mobile business applications to connect to and communicate with business solutions. With this, a Windows Mobile 5.0 or Windows Mobile 6.0 device can exchange data with the ERP server, via the Mobile Server, over various types of networks, including WiFi, GPRS, and EDGE. The architecture supports integration with Microsoft Dynamics AX 4.0 out of the box, and enables integration with other systems as well. The product is designed to:

- Enable users to work in an occasionally connected fashion
- Provide the mobile worker with access to the latest information and software when connected
- Limit the cost of connecting to the network
- Ensure the integrity of data transmissions over the network
- Make it possible to integrate with different business systems
- Minimize the amount of business logic stored on the mobile client.

Architecture Overview

The integration architecture builds on the latest Microsoft technologies and components. The architecture is divided into three tiers: the client, the middle tier, and the server.

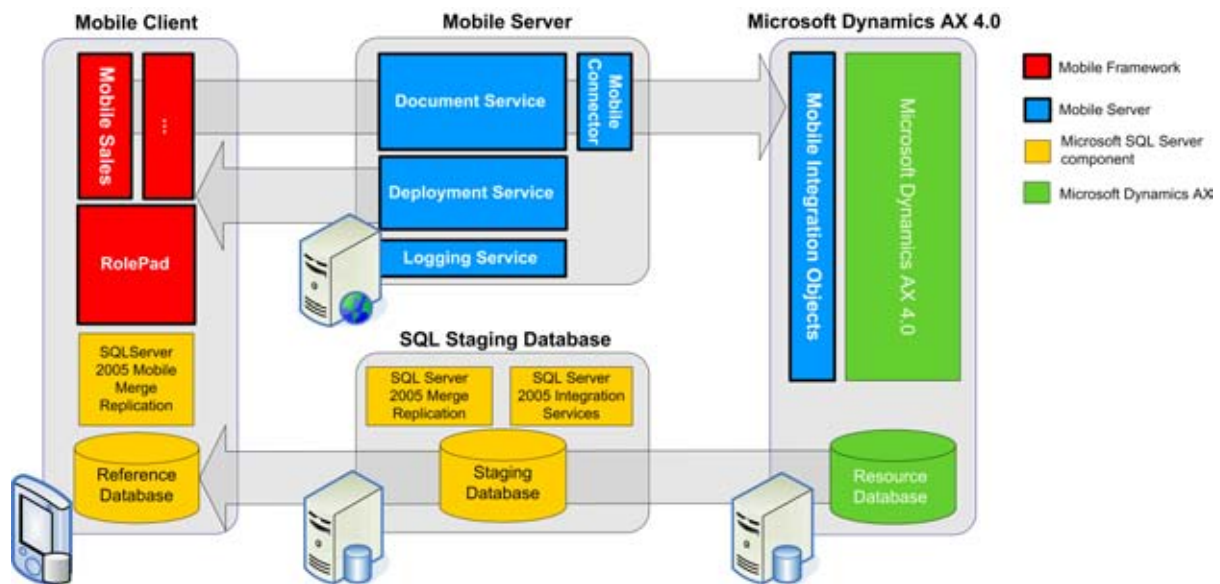


Figure 9: Mobile Framework Architecture

The Client

The Mobile Development Tools support Windows Mobile 5.0 and Windows Mobile 6.0 pocket PC type devices. The mobile client is installed with the following:

- The RolePad mobile application software
- Microsoft SQL Server 2005 Compact Edition
- .NET Compact Framework version 2.0

The database contains a role-specific subset of the data contained in the resource database in the business solution. Having this database on the mobile client makes it faster to access information than going over the network. It also gives users access to information when they are working offline.

The Middle Tier

The middle tier consists of two elements; The Mobile Server and an SQL Server 2005 staging database with related Web synchronization setup. Using Web synchronization for merge replication, data is replicated to the mobile device when the user requests it. Because the synchronization occurs using HTTPS protocol, the Merge Agent on the mobile device just needs to connect to an internet URL that points to the replication components that are installed on a computer running Microsoft Internet Information Services (IIS). These components synchronize the mobile device database with the business solution database. The Internet connection uses secure Sockets Layer (SSL), so a virtual private network (VPN) is not required.

The Mobile Server

The Mobile Server consists of three services; Document, Logging, and Deployment. They are all based on Windows Communication Foundation, and require the .NET Framework version 3.0.

The Document Service

The document service forwards requests sent from the mobile client to the server. It uses a business connector to communicate with the ERP system. This component is a .NET type, implementing a specific interface. The actual implementation of this component will vary from business solution to business solution, depending on the technology used to interface with that system.

The Logging Service

The logging service collects logs generated on mobile devices and stores them in a common repository. Administrators can use a Microsoft Management Console (mmc) snap-in to view and manage logs.

The Deployment Service

The deployment service distributes and updates the mobile application on to user devices. Administrators can use an mmc snap-in to configure and monitor the deployment service.

Staging Database

The staging database is a Microsoft SQL Server 2005 database that contains a subset of the business solution's resource database. Its purpose is to supply mobile devices with the latest data, in a format they can use. Depending on the business solution, data from the resource database might require modification.

The Server

To communicate from the Mobile Server to the business system, integration objects must be deployed on the business system server itself. These objects contain code that is needed to process requests from the mobile device and update the business solution. For Microsoft Dynamics AX, the Mobile Development Tools provide all necessary components. And, the open architecture enables you to develop integration objects for other ERP systems.

Data Transfer to the Client

This next section describes how the components of the Mobile Development Tools work together to exchange data and keep the mobile user and business solution up-to-date.

Getting Data from the Business Solution to the Client

Moving data is a two-step process: The first step is to use Microsoft SQL Server Integration Services to move data from the business solution resource database to the staging database. The process is a simple ETL (Extract, Transform, Load) operation on the subset of data needed by all mobile devices. This extraction is done directly on the database, so the publications configured for the clients, needs to be aware of potential security restrictions set up in the ERP system.

The second step is to use the SQL Merge Replication component of the staging database to publish subsets of data towards the clients. The filtered down version of the staging database is sent to the mobile client, where it is stored in the reference database. The reduction in the size of the data from the resource database to the mobile client can be significant. For example, a typical resource database might be around 40+ GB and the corresponding reference database less than 20 MB. This reduces the bandwidth required for transmitting over the network, which lowers costs.

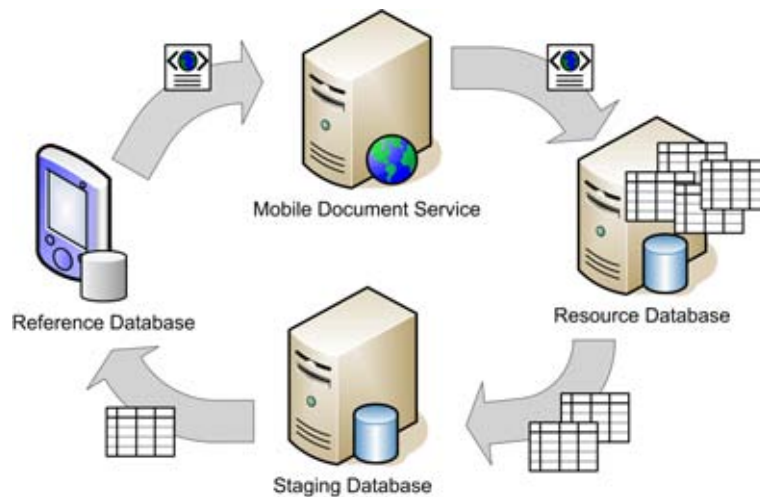


Figure 10: Data Transfer

Sending Data from the Mobile Device to the Business Solution

From the mobile client, data input and transactions to the business solution, such as updating a customer or placing an order, are done using XML. The XML is stored in request documents. Request documents are sent over the Internet through the document service on the Mobile Server. The Mobile Server forwards request documents to the business solution, where they are passed on to the integration objects. These objects then interpret the request documents and process the requests.

Data Transfer Back to the Server

Work captured on the client is kept in request documents. They represent a transaction that can be interpreted and processed by the business solution. Using XML ensures that the correct business logic is applied without having to replicate the business logic on the mobile device.

A request document is made using a combination of orchestration and tasklets. The orchestration defines the request document for a group of tasklets, and the tasklets contribute input to the request document. As a user navigates through an orchestration on a device, each tasklet outputs data that is serialized into an XML block. The individual XML blocks are combined into a single request document that is sent to the business solution for processing. Figure 11 provides a simplified illustration of this process for an orchestration used to create a customer.

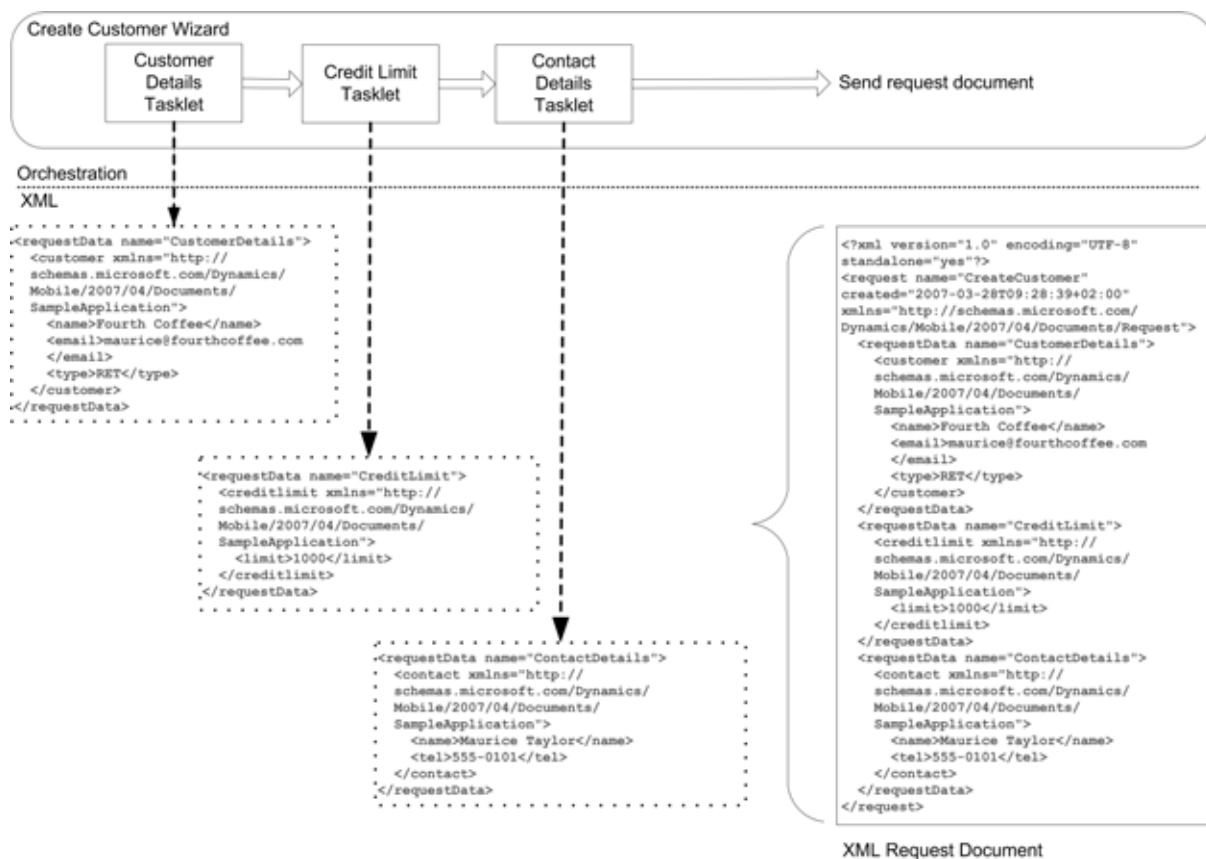


Figure 11: Request Document

Note: Processing a request document requires implementing a document handler to map the input to the proper logic in the business solution.

Dispatching Request Documents and Controlling Costs

When a request document is initiated, it is placed in a queue on the mobile device where it awaits action. If there is a network connection, the request document is dispatched. If there is no network connection, the request document remains in the queue until a connection is available. The request document queue serves two purposes. First, it allows users to make transactions even when offline. These transactions can then be dispatched later on, when a network connection is available. You can also use the queue as a way to control your network costs by limiting the type of request documents dispatched over high-cost networks. Mobile devices support a number of network connection types including EDGE, GPRS, WiFi, and DTPT (Desktop Pass Through). The cost of using these connections varies. EDGE is typically the most expensive and DTPT the least expensive. The Mobile Framework allows you to prioritize when request documents are dispatched based on the type of network connection available and the type of document to be sent.

The client application can subscribe to events on request documents in the queue, so that work can be performed on the client upon successful registration of an order, and so on.

Conclusion

Built on the .NET Compact Framework, Microsoft Dynamics Mobile enables you to quickly create applications that connect mobile workers to their company's ERP system from outside the office. Microsoft Dynamics Mobile includes:

- Mobile Sales application, which enables you to connect to Microsoft Dynamics AX out of the box, but which can be readily extended to support other business solutions and tailored to support specific roles.
- Mobile Framework, which developers can use to quickly create mobile applications from reusable, compiled units called tasklets. And, by making simple modifications to the XML that describes tasklet flow and data transfer, developers can quickly customize or change application behavior, without modifying the tasklet code.
- Mobile Server, which manages connections to the ERP System and synchronizes data between the application and business solution. Mobile Server also provides a logging management console for monitoring, categorizing and prioritizing logs from the mobile device, as well as a deployment management console for installing and updating the mobile application on the device.