

# Windows Phone 7 IN ACTION

Timothy Binkley-Jones  
Massimo Perga  
Michael Sync

MEAP

 MANNING





**MEAP Edition  
Manning Early Access Program  
Windows Phone 7 in Action version 8**

Copyright 2012 Manning Publications

For more information on this and other Manning titles go to  
[www.manning.com](http://www.manning.com)

# Table of Contents

- 1 A new phone, a new operating system***
- 2 Getting started with Windows Phone 7***

## **Part 1 Core Windows Phone**

- 3 Fast Application Switching and Background Agents***
- 4 Launching tasks and choosers***
- 5 Storing data***
- 6 Working with the camera***
- 7 Extending the phone applications***
- 8 Using Sensors***
- 9 Push notifications and network communications***

## **Part 2 Silverlight**

- 10 Silverlight for Windows Phone***
- 11 Building Windows Phone UI with Silverlight controls***
- 12 Manipulating and creating media with MediaElement***
- 13 Using Bing maps and the browser***

## **Part 3 XNA**

- 14 Integrating Silverlight and XNA***
- 15 XNA input handling***

## **Appendixes**

- Appendix A Expression Blend for Windows Phone***
- Appendix B Silverlight primer***
- Appendix C App Hub and Marketplace***

# 1

## *A new phone, a new operating system*

This chapter covers:

- Introducing Windows Phone 7
- Understanding the Hardware
- Porting applications from other platforms
- Developing for Windows Phone

Windows Phone 7 is more than a new operating system. Windows Phone 7 is the operating system, a powerful hardware platform and several web services, all combined to provide a great experience for the busy Life Maximizer. Life Maximizer is the term used by Microsoft to represent the target consumers of the new phone. Life Maximizer's demand the most from their phone as they balance work and life, and use their phone to manage their busy life style. Windows Phone 7 was designed to let users get tasks done faster and allow them to get back to the important aspects of their life.

The Windows Phone 7 operating system is Microsoft's latest entry into the fiercely competitive mobile market. Windows Phone 7 is not an upgrade of previous mobile operating systems, Windows Mobile and Windows Phone 6.5. Microsoft has re-imagined what a mobile operating system should be and completely changed the rules on how to build mobile applications. To power the phone, Microsoft started with familiar foundations in Windows CE, the .NET Compact Framework and the Zune user interface, adapted the Silverlight and XNA libraries, and then added entirely new APIs for interacting with mobile hardware, sensors and software. To enable developers, Microsoft created a toolbox comprised of Visual Studio, Expression Blend and XNA Game Studio.

The first version of the Windows Phone 7 operating system was released in October 2010. Microsoft followed the release with an update in the early months of 2011, adding a copy/paste support and performance improvements. At the Mix 2011 conference, Microsoft unveiled details about Windows Phone 7.5 operating system and the corresponding Windows Phone SDK 7.1. The Windows Phone 7.1 SDK includes several new features including fast application switching, background agents, access to the camera hardware, and a built in SQL CE database engine. Windows Phone 7.5 also exposes new compass, gyro and motion sensors.

**NOTE** We find it a bit confusing that the new operating system is versioned with 7.5 while the corresponding SDK is versioned 7.1. Throughout this book we will refer to both operating system releases as Windows Phone 7 or just Windows Phone. We will provide notes and tips when discussing features that are only available in the Windows 7.1 SDK.

In this chapter we'll present the motivation behind this revolution in the Microsoft OS for mobile devices. We detail how Windows Phone 7 differs from previous mobile operating systems so that you can assess the capabilities of the new platform and understand how existing designs and code can be ported. We describe the minimum hardware specifications common to the different Windows Phone 7 devices so that developers can confidently target equipment that will always be available. Finally, we introduce the developers tools that you will use throughout the book to build applications and games targeted at the Windows Phone.

## ***1.1 Rebooting the Windows Phone platform***

Microsoft has been building operating systems for mobile devices and phones for over a decade. One of the earliest versions was Pocket PC 2000, running on palm-sized devices like the HP Jornada and the Compaq iPaq. These early devices were not smartphones, but were portable computers or PDA's targeted for business users and did not initially include phone hardware or network connectivity. Users interacted with these devices using a stylus on a single-point touch screen and an awkward hardware input panel. Pocket PC 2000 was initially built on Windows CE 3.0, and later added the first version the .NET Compact Framework. Device manufacturers often created custom builds of the operating system tightly coupled to the specific hardware on a single device – making operating system upgrades impossible for most users.

Until Windows Phone 7, the most recent versions of Microsoft's operating system for mobile devices has been Windows Mobile 6 and Windows Phone 6.5. Windows Mobile 6 is built on Windows CE 5 and includes the .NET Compact Framework 2.0 SP1. Windows Mobile 6 comes in three editions – Standard, Professional and Classic.

**NOTE** For the remainder of the book, when the term Windows Phone is used without a version number, we are referring to Windows Phone 7.5. We will use Windows Mobile or Windows Phone 6.5 to refer to older versions of the phone operating system.

Mobile phones have had a rapid and incredible evolution in the past several years. Once intended solely for business users, mobile phones are now predominately consumer devices, and in many cases have replaced land-line services as a user's only phone. Smartphones now include radios, music players, cameras, global positioning systems, compasses and accelerometers. Single-point touch screens that required a stylus have been replaced with multi-point touch screens that work with your fingertips. Awkward hardware input panels have been replaced with software input panels and optional hardware keypads.

Apple led the smartphone revolution with the release of the iPhone in June 2007, and the introduction of the App Store in July of 2008. Google followed with the introduction of the Android OS and Market in October of 2008. Since then, Microsoft has seen declines in Windows Mobile's market share as consumers and device manufacturers turn to smartphones running new mobile operating systems.

But phone hardware and mobile operating systems are not all that have changed in the last decade. It is now an online world where users are in nearly constant contact with friends, co-workers, family, that high school buddy they haven't seen in 20 years, and random followers they have never met. Applications that once worked only with local copies of documents and data are now interacting with services running in the cloud. And with all this online presence and exposure, security is extremely important. It is no longer acceptable to give software full access to hardware, or to the data stored in the file system.

Application development platforms and paradigms have changed as well. With the rise of web applications, a whole new style of application development came into power. Rich interactive applications are the norm, complete with animations, dynamic transitions and cool graphics. User interfaces are no longer built by developers, but are created by designers who use a whole different set of tools.

Microsoft set out to build a new Windows Phone operating system designed to meet the demands of the altered smartphone market. Microsoft realized they would need a new operating system, backed by a reliable hardware platform, to compete with Apple and Android.

## **1.2 Windows Phone foundations**

Every application developer must understand the hardware and software platforms where his code will run. This is true if you are building desktop applications, web services or mobile applications. When building Windows Phone applications, you should understand the hardware specifications and should know how much memory you can expect to be installed and the supported screen resolutions. Windows Phone provides a unique look and feel which developers should respect when designing user interfaces. You should also know how to leverage or extend the features of built-in applications and services. In this section we talk

about the Windows Phone hardware specifications, user interface look and feel, native applications and the platform APIs you will use to build your own applications.

### **1.2.1 Hardware specs**

With the redesign of the operating system, Microsoft has taken this opportunity to define clear hardware specifications for Windows Phone 7 devices. All devices must meet the minimum hardware requirements.

On Windows Phone 7, all the devices have the same screen resolution of 800x480 pixels. The physical screen dimensions will also be very similar among across all devices. A common screen size and resolution allows the same user interface to be reused across the different Windows Phone devices.

All the Windows Phone devices will provide the user a full four point multi-touch experience. The operating system provides a software-based input panel (SIP) to enable text input for devices without a physical keyboard. Of course, phone manufacturers can add additional user inputs mechanisms, such as a landscape or portrait physical keyboard, but extra hardware will not be allowed to add extra features to the standard typing. The touch screen is capacitive to give the best experience possible on a mobile device.

Windows Phone 7 devices come with an accelerometer, a compass and an optional gyroscope. Developers access the raw data from each sensor or use the motion sensor APIs, which wrap up all three sensors into a simple to use library. The operating system detects when a device has been rotated from portrait to landscape orientation. The sensors can also be used to as an input mechanism for controlling an application or game. The sensors are covered in more detail in chapters eight and fifteen.

An FM Radio is a mandatory requirement for Windows Phone 7 devices. A user can access the radio from the Zune application in the Music + Videos Hub, but developers can also create a customizable FM radio player using the `FMRadio` class in the `Microsoft.Devices.Radio` namespace. Programming the FM radio is demonstrated in chapter seven.

The minimum hardware specifications also require:

- An Assisted GPS receiver to enable location-aware applications.
- A camera having a minimal resolution of 5 Megapixels.
- A GPU supporting DirectX 9 acceleration
- Either a 800 MHz or a 1 GHz ARMv7 CPU
- A minimum of 256 MB of RAM and 8 GB of Flash storage

The Windows Phone hardware specification requires certain hardware buttons to be present. Many of these keys are not exposed to developers, and applications cannot detect when they are pressed. The physical buttons which will be mandatory for all Windows Phone devices will be:

- Volume Up

- Volume Down
- Back
- Start
- Search
- Camera
- Power On / Off

A minimum hardware specification has simplified the task of developing a Windows Phone application. These common hardware specifications have allowed Microsoft to create an emulator which will cover most of the possible user interactions with the device, so that you can test most experiences in your emulator.

Microsoft defined a clear hardware specification to ensure users and developers have the same experience on every device. Microsoft also designed a new user interface to provide a clean look and feel.

### **1.2.2 A New User Interface**

Windows Phone has a completely redesigned the user interface moving from an icon-centric style to the new graphical interface previously developed for the Zune HD media player. Microsoft designers spent some time looking for a proper way to present content and realized a very intuitive style already existed. Signage and typography in railway or metro stations, shown in figure 1.1, are in fact concise ways to present information to people coming from different cultures. Why not port this concept to Windows Phone?



Figure 1.1 Common signs in railways and airports. On the left we have icons integrated by text; whilst on the right only icons are used.

The second pillar of the user interface is full-touch support. The successes of devices implementing a full-touch user interface are due to the immediacy provided by a natural way of interacting with applications. Concise indications and full-touch come to play an important role in development of our applications as we must align to these concepts when we design our user interface.

One well-known defect of the applications written for Windows Mobile was the lack of a common user experience. We have seen applications very aligned to the template generated by Visual Studio but implemented with a user interface that was built to match the iPhone user experience. This is confusing to the user, and you should make every effort to match your creations to the Metro design language adopted by the native Windows Phone applications.

Last but not least, when developing your application you want to target as many users or customers as possible. Globalizing an application doesn't mean just to make it right in terms of functionality, but also in terms of contents. We strongly recommend avoiding expressions or icons which do not have a global meaning. Also remember that your application will be inspected by Microsoft prior to publication it to the Marketplace. There are Marketplace guidelines about the content which can and cannot be presented through a Windows Phone application.

### **1.2.3 User Experience**

Understanding the user experience of the Windows Phone is important to building an application that feels like it belongs on the phone. The built in applications, called hubs, establish the look and feel of the device and provide integration and extensibility points for third party applications.

**NOTE** only the start experience and the application list are accessible on the emulator.

The hubs are built with two new UI controls named Panorama and Pivot. You can read more about using the Silverlight version of the Panorama and Pivot in chapter ten.

#### **START EXPERIENCE**

The Start Experience is the home screen for Windows Phone. It is the screen displayed when the phone is started. When the user presses the windows button, they are brought back to the start screen. A user can pin their favorite applications, games and contacts to the start screen so that they can launch them quickly.

The images displayed on the start screen are named Tiles. Tiles can be dynamic, displaying information relevant to an application. The tile for the Weather Channel application updates with the latest weather conditions. Other tiles are badged when notifications are ready to be viewed. The tiles for email display a count of new mails messages. The image and title that appear in the start screen here are provided by the developer.

Applications can pin multiple tiles to the start screen, each launching to a different spot within the application. Tiles can be updated from code running on the phone, or remotely using the Microsoft Push Notification Service. See chapter two and chapter nine for more details on live tiles.

### **APPLICATION LIST**

The Application List is where all native and third party applications appear. It does not matter if the application is built using Silverlight, XNA or is a native application built by Microsoft, the device vendor or the mobile carrier. The developer determines the application title and icon that are shown the application list. Games are not listed in the application list.

### **GAMES HUB**

If your project is declared to be a game, it will be listed in the Games Hub instead of the Application List. The Games Hub is divided into several areas:

- The Collection view lists the games installed on the device
- The Spotlight view displays news from Xbox Live
- The Xbox live view provides access to the users Xbox Live gamer profile
- The Requests view lists Xbox Live invitations, messages and notifications

The game title and icons displayed in the collection are declared by the game developer.

### **MUSIC + VIDEO HUB**

The Music + Video hub is the central place where you can find all music, video and podcast activity on the device. The Music + Videos hub is divided into four areas:

- Zune is the central point for playing music, videos, podcasts, radio, and Zune Marketplace.
- History contains the list of music, videos, playlists, artists, podcasts, FM radio stations that you recently played. This includes media played by third party applications that integrate with the hub.
- New contains the list of new music, videos, or podcasts that you synced to the phone or downloaded from Zune Marketplace. Third Party applications can add items to the New view.
- Apps contains the list of Music + Videos hub applications that are installed on the device. Third Party media applications are listed here.

The Music + Video Hub provides a few integration points to third party applications. You can read more about the Music + Video Hub in chapter seven.

### **PICTURES HUB**

The pictures hub is the place where you can see all of your photos from difference sources. All photos that you took with your mobile, synced from the computer, downloaded from the Internet or opened in email will be included in the picture hub. The picture hub is integrated with Windows Live and Facebook, and all photos that you uploaded to those websites will be displayed in picture hub as well. It also shows the latest photos of your friends in Facebook as well.

The picture hub can be extended by third party applications that implement phone editing or sharing features. Extending the picture hub is described in chapter seven.

## PEOPLE HUB

The People Hub is the contacts application for Windows Phone. Here is where you find the list of contacts, and their phone numbers and addresses. The People Hub also displays the latest status and activity obtained from Windows Live and Facebook. Third party applications can read data directly from contacts database, and can read and write contacts data Launchers and Choosers, which are introduced in the next section.

Unlike the other hubs, the people hub is not extensible by Windows Phone applications. The people hub can be extended by registering new activity streams with the user's Windows Live account. Activity streams, a format for syndicating data from social networking applications, are beyond the scope of this book. You can read more about activity streams by visiting <http://activitystrea.ms>.

Understanding Windows Phones Hubs and how they can be extended is key for building applications that enhance user productivity and are integrated with the operating system. Third party integrated applications and extensions build on top of the features exposed in the platform APIs and frameworks.

### 1.2.4 Platform APIs and Frameworks

An application runs in a sandbox and cannot use native APIs, communicate with other processes, or read from the file system. These security measures limit the ability to integrate with native applications and databases. To ease these limitations, native applications also expose various integration points. These integration points come in the form of launchers, choosers and extensions. The platform also provides access to network APIs so that applications can use web services external to the device. Finally, facilities such as location and notification services are available to third party developers.

## LAUNCHERS

Launchers allow your code to activate a native or built-in application. Data can be passed to the launched application. When the native application is launched, your application is deactivated. Launchers are provided to activate the phone dialer, media player, web browser and other native applications. Launchers are the only way to initiate a phone call or send an SMS. Launchers are covered in depth in chapter four.

## CHOOSERS

Choosers return data to an application. Choosers are provided to retrieve email addresses, phone numbers, physical addresses and photographs. Choosers also launch a native application, resulting in the deactivation and/or termination of your application. Choosers are also covered in chapter four.

## EXTENSIONS

Extensions allow an application to integrate their features seamlessly into a native application. For example, the Picture Hub allows photo editing applications be launched from its Apps list and from the share and apps menus. The Music + Video Hub allow applications to appear in its Apps list.

## NETWORKING

Windows Phone provides HTTP and sockets network communication. HTTP communication is implemented in the `WebClient`, `HttpWebRequest` and `HttpWebResponse` classes found in the `System.Net` namespace. TCP and UDP communications is implemented with the `Socket` class in the `System.Net.Sockets` namespace. Networking is covered in depth in chapter nine.

## NOTIFICATIONS

The Microsoft Push Notification Service provides an API where a phone user can subscribe to a set of custom events. The notification events are defined by 3rd party applications and must be sent from a dedicated web service implemented by the application developer. Notifications are displayed to the phone user either on the application's live tile in the start experience, at the top of the screen as a toast notification, or with-in the running application. We show you how to build a notification application in chapter nine.

## LOCATION

The Location service uses data from the wireless and cellular networks and GPS to allow you to create location-aware applications. Calls to the location cloud service are abstracted behind the `GeoCoordinateWatcher` class in the `System.Device.Location` namespace. In chapter eight we show you how to use `GeoCoordinateWatcher`.

## CUSTOM WEB SERVICES

Beyond providing access to business application data or social networks, custom web services can be used to overcome some of the limitations of phone. If you have a suite of applications that share data, you can use a web service to share the data between them.

### 1.2.5 AppHub and the Windows Phone Marketplace

AppHub is the portal where Windows Phone and Xbox LIVE Indie Game developers can find the tools and resources for building and selling applications and games. The AppHub is where you can download the developer tools. You can also find sample code, tutorials and documentation. If you need advice on a tricky problem, you can submit a question to the developer forums on the AppHub. The AppHub is located at <http://create.msdn.com>.

Before you can deploy and debug your application on a real phone, or publish your application to the Windows Phone Marketplace, you must purchase a yearly subscription to the AppHub. Depending on what you are building, you might consider waiting to purchase an AppHub subscription until your application is nearly complete, using the emulator to build and test your application.

**TIP** College students receive free AppHub subscriptions through the DreamSpark program. DreamSpark is a Microsoft program providing students with free copies of retail development tools and servers. You can learn more about DreamSpark at <http://dreamspark.com>

Once the application has been developed, it must go through an approval process run by Microsoft before being published to the Windows Phone Marketplace. This will ensure that the application conforms to Microsoft requirements for a Windows Phone 7 application. Microsoft's requirements are detailed in the document *Windows Phone 7 Application Certification Requirements* available from the AppHub and MSDN. More details about marketplace registration is provided in the appendix.

### **1.3 Comparing Windows Phone to other mobile platforms**

This book is written primarily for developers who have some experience working with C# and Silverlight. We focus on the features and APIs that have been introduced specifically for the phone, or have been modified to fit the phone's unique characteristics.

If you already use Silverlight to develop applications, then you know it has matured rapidly over the last few years. Silverlight's success as a light weight application framework makes it ideal to use as the application framework on the mobile device. The Silverlight Framework is rich in features and has been proven with browser and desktop applications. You will find many of the familiar features and tools. The Windows Phone version of Silverlight is version 4.

**NOTE** The initial version of Windows Phone 7 used Silverlight 3. Silverlight 4 shipped with Windows Phone 7.5.

If you have used XNA Game Studio, then you know that XNA is built to run on the Xbox, Windows and the Zune – Windows Phone is just one more platform. Existing developers can easily build and port games for the new devices. Windows Phone introduces a new game development model by integrating Silverlight with XNA which we introduce in the final section of the book.

If you are not already a Silverlight developer, do not despair. The appendix includes a quick primer for Silverlight and Manning has published several books on C# and Silverlight, which you can find at <http://www.manning.com/catalog/dotnet>.

But what if you are coming to Windows Phone from some other background? How does the Windows Phone differ from Windows Forms on Windows Mobile? Where do you begin when porting your iOS or Android application? In this section we try to get you started with Windows Phone development by identifying the similarities and differences with other application platforms.

#### **1.3.1 Windows Mobile**

If you are a third party Windows Mobile developer, then you should know that Windows Phone 7 is not Windows Mobile. You cannot use C++ or the Win32 API. If you were thinking that Windows Phone 7 would be backward compatible with Windows Mobile, then you are out of luck. You may have heard that there is a native SDK, but for now, only device manufacturers, mobile operators and other special partners get to use it.

Windows Mobile has been a very popular operating system because of its extreme customization. Windows Phone is a new operating system and not an upgrade, and applications written for Windows Mobile and Windows Phone 6.5 are not compatible with Windows Phone 7. Windows Mobile development environments and tools are also incompatible. In this section we will illustrate the major changes which will impact every developer with previous experience in Windows Mobile development, starting with the user interface.

### **BUILDING YOUR INTERFACE**

Windows Mobile applications are built with C/C++ and low-level API calls. Neither of these options is available to the Windows Phone developer, who must now use Silverlight and XAML. XAML is a user interface design language first introduced with the Windows Presentation Foundation (WPF) and is a core component of Silverlight. XAML enables separation between user interface and the code that implements application logic.

### **DRAWING ON THE SCREEN**

Windows Mobile provided two native APIs for drawing text and graphics to the screen:

- Graphics Device Interface (GDI)
- DirectX

Both the APIs are quite low level and have a steep learning curve for the standard developer. Being native libraries, neither GDI nor DirectX can be called from managed code running on Windows Phone. The XNA Framework is the managed alternative to DirectX, implementing many of the features available in the DirectX libraries. Silverlight makes use of DirectX and your application will be hardware accelerated behind of the scenes.

### **CHANGES IN THE USER EXPERIENCE**

The Today Screen has been the traditional Windows Mobile shell or system UI. Windows Mobile allows the system shell to be replaced by custom user interfaces built by device manufacturers and third party developers. Windows Phone provides a new simplified user interface that cannot be replaced or modified. The simplified user interface has removed some traditional controls, whilst it has introduced new ones designed for touch interaction and to simplify creating user interfaces.

### **SOFT KEYS SUPPORT**

One change you need to keep in mind if you're porting a Windows Mobile application to Windows Phone is the full lack of soft keys, including the hardware buttons associated with them. Another change in the user interface are the menus; they're now basic and most of them are no more than a list.

### **CHANGES IN THE API**

The biggest strength of Windows Mobile was probably its large compatibility in terms of programming paradigm and APIs with Windows desktop. This meant that every Windows desktop developer was a potential Windows Mobile developer. On the other hand, Windows Mobile compatibility with the Win32 API brought an additional complexity to the application.

## **MEMORY MANAGEMENT**

A major problem with Windows Mobile applications was the possibility of memory leaks. Because C/C++ requires code to manage its own memory, if the developer allocates memory but forgets to release it during the execution, memory is lost until the process is terminated. Managed applications written in C# or Visual Basic use the .NET Compact Framework's garbage collector, which is an invisible helper taking care of memory management.

## **ACCESS TO THE FILE SYSTEM**

Windows Mobile applications have almost full access to all the files available on the file system. This capability is very useful when developing document centric applications such as, a text editor, so that the user will be able to open a file on the file system regardless of its location. On the other hand, a malicious application could corrupt the file system and prevent other applications from being executed, or sniff out sensitive data.

For this reason, each Windows Phone application is locked into a sandbox and can only access files in a reserved portion of persistent memory named isolated storage. There is no way for a Windows Phone application to access data contained in isolated storage belonging to a different application. Isolated storage is covered in the chapter five. Applications requiring access to the whole file system cannot be developed under Windows Phone 7.

## **MULTITASKING**

The Inter-Process Communication (IPC) API of Windows Mobile allows different processes to synchronize with each other using the operating system primitives. Sometimes this was useful as Windows Mobile is a multitasking operating system.

Windows Phone does not support true multitasking, at least for applications developed in XNA or Silverlight. Fast application switching allows multiple applications to be resident in memory, but only the foreground application is running, with the background applications remaining in a dormant state. Applications can use background agents to perform limited types of work when an application is not in the foreground. Fast application switching and background agents are described in chapter three.

One new possibility for mobile developers previously available only to desktop developers is the thread pool. As the creation of a thread is a quite expensive process and most of the threads are usually blocked on some event, a set of threads is provided by the operating system which will be automatically re-used during the execution. All this is provided for free by the system; in addition to being very easy to use, it's a good practice when designing for new systems which could embed multi-core processors. Thread pool automatically scales to multi-core processors without need of code rework.

As you can see, Windows Phone 7 is a completely different platform from Windows Mobile 6. The work required to port exiting Windows Mobile applications is no different from that required to port iOS or Android applications.

### **1.3.2 Apple iOS**

At first glance, you might think there is very little in common when developing applications for an iOS device and the Windows Phone. On one platform you use Objective-C to write

native applications, on the other you use C# to write managed applications. It is our opinion that a programming languages and frameworks are just tools in the tool belt of a developer, and good developers make use of several languages and frameworks. If you look beyond the languages and development environments, many of the fundamental concepts exist on both platforms.

Apple and Microsoft both provide free development tools complete with device simulators. Each platform has a set of style guides that applications should adhere to. Each platform requires a fee-based subscription in order to deploy an application to an actual device. Each platform has a certification process and application store.

### **BUILDING YOUR INTERFACE**

One thing to keep in mind when porting an iOS application is the differences in the user interface guidelines. You should not build an application with an iOS look and feel for the Windows Phone. An iOS application ported to Windows Phone will have a different look and feel, user interaction model and workflow. Do not use chrome and icons from iOS.

Is your application built with controls from UIKit or does it use OpenGL ES? The Silverlight Framework offers many of the controls and widgets provided by UIKit. On the other hand, OpenGL developers will use the XNA Framework to build applications. You can also mix application style widgets from Silverlight with XNA type graphics.

You will build your Silverlight applications using Visual Studio and Expression Blend. Your Views will be built using XAML, an XML based markup language. XAML can be coded by hand in Visual Studio's text editor, or with the visual editors in Visual Studio and Expression Blend. The core Silverlight Framework, along with the Silverlight Toolkit, provides most of the controls you will need when building an application.

If your iOS application uses CoreAnimation, you will use the animation and storyboard classes from the `System.Windows.Media.Animation` namespace. Learn to use Expression Blend's storyboard editor if you are doing anything beyond very simple animations.

Silverlight applications are navigation style applications, driven by the `NavigationService`. The `NavigationService` is similar to the `UINavigationController` provided by the iOS framework, and is used to move between different pages or views. The difference is that all Silverlight applications use the `NavigationService`, even the simplest one page application.

### **INTERACTING WITH THE NATIVE APPLICATIONS**

Like the iOS SDK, Windows Phone provides limited access to the phone dialer, SMS text application, and email. On iOS, the phone dialer is accessed via the tel URL; on Windows Phone you use the `PhoneCallTask`. `MFMessageComposeViewController` and `MFMailComposeViewController` are replaced by `SmsComposeTask` and `EmailComposeTask`.

The iOS SDK provides access to the address book with several classes in the `Address Book` and `Address Book UI` frameworks. On Windows Phone, read only access to the address

book is exposed via classes in the `Microsoft.Phone.UserData` namespace. Developers can also interact with the contacts database via a few launchers and choosers. You can prompt the user to choose a phone number, email address or physical address with `PhoneNumberChooserTask`, `EmailAddressChooserTask`, and `AddressChooserTask`. You can prompt the user to save a phone number or email address with `SavePhoneNumberTask` and `SaveEmailAddressTask`. You can read more about launchers and chooser in chapter four.

### USING THE SENSORS

Like the iPhone, the Windows Phone has an accelerometer, a compass and a camera. Some Windows Phones will also have a gyroscope. The initial release of Windows Phone did not provide an API to access the compass, and access to the camera was limited. The Windows Phone SKD 7.1 introduced new APIs providing access to the compass, gyroscope and the camera. Using the `CameraCaptureTask`, you can launch the camera UI and manipulate a photo taken by the user. You can take direct control of the camera either using the `PhotoCamera` or the `WebCamera` APIs. Working with the camera is covered in chapter six.

The Windows Phone complement to `UIAccelerometer` is the `Microsoft.Devices.Accelerometer` class. The `Compass` class is the Windows Phone equivalent to `CLHeading`. Motion detection features available by the Core Motion framework are provided by the `Gyroscope` and `MotionSensor` classes. We show you how to use the accelerometer, compass, and gyroscope in chapter eight.

### STORING DATA

An iOS application can store its data in user defaults, on the file system, or in a database. The iOS SDK makes use of SQLite for local database management.

Windows Phone does provide limited access to the file system. An application can only write files to isolated storage, and it does not have access to any other part of the file system. Isolated storage is similar to an iOS application's Documents folder.

Another way to store data is with the `IsolatedStorageSettings` class. This class is similar to the `NSUserDefaults` class in the iOS framework. It is intended to be used to store lightweight data objects and is ideal for storing user preferences. One difference between `NSUserDefaults` and `IsolatedStorageSettings` is that `IsolatedStorageSettings` is not global, and settings cannot be shared between different applications.

Applications can store data in a Microsoft SQL Server Compact (SQL CE) database using the Linq to SQL framework. SQL CE is a lightweight database engine designed to run on mobile devices. The database files are written to a special folder in isolated storage, and cannot be shared with other applications. Chapter five demonstrates how to use each of the data storage options in your applications.

### MEDIA

The iPhone uses the iPod software to play audio and video files. The iOS SDK's Media Player framework allows developers to access the library of music and videos, and to play them

inside their applications. The Windows Phone uses Zune for its media library, shown to users in the Music + Videos Hub. Applications can play audio and video files with the `MediaPlayerLauncher` class. Developers can also access the Zune library using the classes in the `Microsoft.Xna.Framework.Media` namespace. The `MediaPlayer` class can be used to play songs, while the videos are played with the `VideoPlayer` class.

Silverlight applications can use the XNA Media framework, but Silverlight also has its own media controls in the `System.Windows.Media` namespace. The `MediaElement` control supports audio and video playback. The `MediaStreamSource` class can be used to manipulate audio and video playback or implement custom media containers.

The Windows Phone equivalent to the iOS's `AVAudioRecorder` class is the `Microsoft.Xna.Framework.Audio.Microphone` class.

Your application can integrate into the Music + Video Hub on the phone. Your application can be listed in the hub's Apps list, and media played by your application can be shown in the Hub's History page.

You can read about working with Media, the Microphone and the Music + Videos Hub in chapters seven and twelve.

## **NETWORKING**

The iOS SDK offers several classes to enable network programming. A developer can choose to program using raw sockets, or higher level protocols such as HTTP and FTP. Windows Phone offers sockets and HTTP support. You perform HTTP communication using the `HttpWebRequest`, `HttpWebResponse` and `WebClient` classes in the `System.Net` namespace. Sockets programming is performed using classes in the `System.Net.Sockets` namespace.

Microsoft has also built a notification service to allow web services to push notifications to a phone. Developers host their own web service or other application. The application service sends notifications to Microsoft's Push Notification web service, which forwards notification to a user's phone. Interaction with the notification service is covered in chapter nine.

As you can see, there are many differences between the iOS and the Windows Phone. There are also a number of similarities and developers should be able to port most applications to the Windows Phone.

### **1.3.3 Android**

Android is another new mobile operating system that is capturing the hearts and minds of consumers and developers. Like the iPhone, there are many differences and many similarities between Android and the Windows Phone. Like Windows Phone, Android runs on a number of different devices, from a number of different manufacturers. Unlike Microsoft, Google has not dictated the hardware specifications to the manufacturers and developers must design and test on several hardware configurations.

Android and Microsoft both provide free development tools complete with device emulators. However, Microsoft requires a fee-based subscription in order to deploy an

application to an actual device and certifies each application before making the application available in the application store.

#### **RUNTIME ENVIRONMENT**

Windows Phone applications run in the .NET Compact Framework Common Language Runtime (CLR). The CLR is a virtual machine much like the Dalvik virtual machine that runs on Android. Applications are packaged in .xap files, which is a zip archive of the assemblies and resources in the application bundle.

The Windows Phones places restrictions on the types of applications that can run on the phone. Android allows for background services and UI-less broadcast receivers to run on the phone. While Windows Phone offers limited support for background operations with background agents, there is no counterpart to broadcast receivers. Windows Phone does not have system alarms or triggers that can directly start an idle application. Windows Phone applications can be started when the user responds to alarms, reminders or notifications.

The Android runtime does limit access to certain features with manifest permissions. Windows Phone uses a similar security model by requiring capabilities to be declared in the application manifest.

#### **BUILDING YOUR INTERFACE**

Android activities are loosely related to pages in a Silverlight application. Each page of an application has a unique address, and the operating system will use a page's URL to navigate to the page when restarting an application. Developers can use a page's URL when creating live tiles. Android programmers declare user interface with layout XML files. Silverlight user interfaces are declared using XAML, which are also XML files. If your application makes use of the Android `MapView`, you will want to read about using the Bing Maps control in chapter thirteen.

#### **INTERACTIONS WITH OTHER APPLICATIONS**

Android applications interact with built-in and third party applications by dispatching Intents. Windows Phone applications interact with native applications via Launchers and Choosers. Windows Phone does not allow third party applications to interact with other third party applications, and developers cannot create new launchers or choosers.

Android applications can replace, enhance or just eaves drop on another application by handling the same Intents. Windows Phone does not allow third party applications to replace any launchers or choosers. You can enhance the Pictures Hub and the Music + Video Hub by implementing the required extensibility points.

Android applications share data by exposing and using content providers. On Windows Phone, there is no way to expose your data to other applications, and other applications cannot use your data.

You can read about the available launchers and choosers in chapter four.

#### **STORING DATA**

An Android application can store its data in shared preferences, on the file system, or in a database. Android uses SQLite for local database management.

Windows Phone does provide limited access to the file system. An application can only write files to isolated storage, and does not have access to any other part of the file system. You cannot read another application's files, and other applications cannot read your application's files.

Another way to store data is with the `IsolatedStorageSettings` class. This class is similar to `SharedPreferences` in the Android framework. It is intended to be used to store lightweight data objects and is ideal for storing user preferences. One difference between `SharedPreferences` and `IsolatedStorageSettings` is that `IsolatedStorageSettings` is not global, and settings cannot be shared between different applications.

Windows Phone applications can store data in a Microsoft SQL Server Compact (SQL CE) database using the Linq to SQL framework. SQL CE is a lightweight database engine designed to run on mobile devices. The database files are written to a special folder in isolated storage, and cannot be shared with other applications. Chapter five demonstrates how to use each of the data storage options in your applications.

## **MEDIA**

Android uses the OpenCORE library to play and record audio files and to play video files. OpenCORE's `MediaPlayer` class is used to play audio, while the `VideoView` widget is used to play video. Windows Phone applications use the `MediaPlayerLauncher` class to play audio and video files. Developers can also access the Zune library using the classes in the `Microsoft.Xna.Framework.Media` namespace. The `MediaPlayer` class can be used to play songs, while the videos are played with the `VideoPlayer` class.

Silverlight applications can use the XNA Media framework, but Silverlight also has its own media controls in the `System.Windows.Media` namespace. The `MediaElement` control supports audio and video playback. The `MediaStreamSource` class can be used to manipulate audio and video playback or implement custom media containers.

The Windows Phone equivalent to the Android's `MediaRecorder` class is the `Microsoft.Xna.Framework.Audio.Microphone` class. You can read about working with media, the microphone and the Music + Videos Hub in chapters seven and twelve.

## **NETWORKING**

Android provides a variety of networking options starting with raw sockets and extending through HTTP. Windows Phone offers sockets and HTTP support. You perform HTTP communication using the `HttpWebRequest`, `HttpWebResponse` and `WebClient` classes in the `System.Net` namespace. Sockets programming is performed using classes in the `System.Net.Sockets` namespace.

Android networking applications can use the `ConnectivityManager` class to determine the status of the device's network connection. To check the network status of a Windows Phone, you use the `NetworkInterface` class in the `Microsoft.Net.NetworkInformation` namespace.

In many ways, the Android platform is more like the Windows Mobile 6 platform. Applications have fewer restrictions and can replace core features of the operating system. Manufacturers can change the look and feel of the operating system. Developer must build for a wider range of hardware configurations. There are going to be a certain set of applications that cannot be ported to Windows Phone because of the limitations enforced by the operating system.

## **1.4 The Windows Phone Developer Tools**

In order to build great applications, you need great development tools. Microsoft's Visual Studio and Expression Blend fit the description. Visual Studio 2010 Express for Windows Phones joins the list of no-cost express developer tools provided by Microsoft. XNA Game Studio has been updated to build Windows Phone games. And a no-cost version of Expression Blend 4 has been made available. All of these tools have been packaged up together and are distributed as The Windows Phone Developer Tools which can be freely downloaded from the AppHub at <http://create.msdn.com>.

### **1.4.1 Visual Studio for Windows Phone**

The Windows Phone Developer Tools installs an express edition of Visual Studio 2010 configured with the phone development tools. If you already have a retail edition of Visual Studio 2010 installed on your computer, the phone development tools will be installed as a plug-in to the IDE. Windows Phone projects can be written in both C# and Visual Basic.

We will use the express edition throughout the book for the screen shots and sample code. Code and user interface design features will work the same in the retail editions of Visual Studio 2010.

You can launch the IDE by opening the Start Menu and clicking on Microsoft Visual Studio 2010 Express for Windows Phone in the Microsoft Visual Studio Express folder. Figure 1.2 shows the Visual Studio IDE.

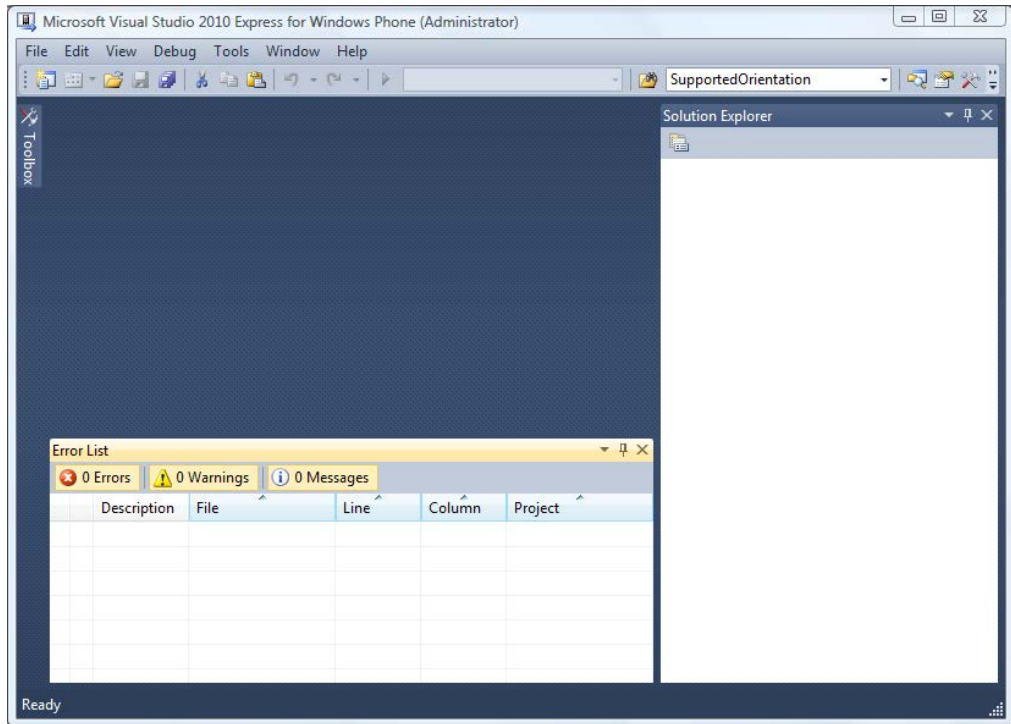


Figure 1.2 Visual Studio 2010 Express for Windows Phone

Visual Studio 2010 Express for Windows Phone, from here on referred to as Visual Studio, can be used in two different modes - Basic and Expert. We suggest you enable expert mode so that you can use all the available features. To enable the expert mode you have to select Tools->Settings->Expert Settings menu option. Expert mode unlocks some toolbars and several menu items.

### 1.4.2 Expression Blend for Windows Phone

Visual Studio has cool features but it's not so friendly for the user interface designers on your team. Microsoft has created a tool for designers named Expression Blend. Originally part of the Expression Studio suite, a no-cost edition of Expression Blend has been provided for creating Windows Phone applications. Expression Blend allows the designer to create user interfaces without writing a single line of code.

Expression Blend can create the same Silverlight projects as Visual Studio. A designer can edit the same solution, project and code files that a developer edits in Visual Studio. While we may occasionally cover Expression Blend features in the book, our focus will remain on using Visual Studio. A primer on Expression Blend is available in the appendix.

### 1.4.3 XNA Game Studio

XNA Game Studio, another Visual Studio add-on, provides a set of tools and libraries that can be used to build games for Windows, Xbox, Zune and now the Windows Phone. XNA Game Studio 4 provides the tools necessary for creating Windows and Xbox games, which are beyond the scope of this book.

XNA applications usually import content or assets created by artists. This could be graphics, 3D models, music or videos. In the last section of this book, we show you how to build Rich Graphics Applications by integrating XNA with Silverlight.

### 1.4.4 Windows Phone Emulator

The Windows Phone 7 Emulator should work on most recent computers. The emulator performs better if your computer has a CPU with virtualization extensions like most of the recent AMD and Intel CPUs. The emulator works best if a DirectX 10 or later graphics card with a WDDM 1.1 driver is present.

**NOTE** You can determine if your computer has a supported GPU and driver with the DirectX Diagnostics Tool that is part of the DirectX SDK. You can download the DirectX SDK from the DirectX Developer Center at <http://msdn.microsoft.com/en-us/directx>

The DirectX 10 GPU and WDDM driver are mandatory for XNA games, but are not necessarily required by most Silverlight applications.

The emulator does not require special binaries to execute XNA games or Silverlight applications for Windows Phone. The emulator can be used to verify orientation changes in your application by using the rotation buttons on the emulator's command toolbar.

Keep in mind that on the emulator you're sharing the network connection of your PC, so the bandwidth available is greater than what would be available to a real phone. The emulator doesn't allow you to simulate out of coverage scenarios, bandwidth changes (i.e. 2.5G to 3G) and when checking network information the `NetworkInterface` class always returns `WiFi`. In order to verify network connectivity, you can use the full working versions of Internet Explorer available in the emulator.

The Settings application found in the application list can be used to change the emulator's default configuration. However, the settings revert to their defaults when the emulator is stopped and restarted. You will need to change the settings to verify your application behave appropriate under different configurations and locales. The emulator settings are:

- Theme and accent color
- Date and Time
- Region and Language

If your computer is running Windows 7 and uses a true multi-touch monitor, the emulator will register touches to the computer's monitor. Otherwise, the emulator simulates touches

with the mouse. The emulator can also switch between using the SIP and treating your computer's keyboard as a hardware keyboard.

#### **1.4.5 Windows Phone Developer Registration Tool**

Applications can only be installed onto a phone by the Windows Phone Marketplace. Limited exceptions are made to phones registered to developers who have accounts with the AppHub. AppHub accounts are not free, and can be purchased from the developer portal at <http://create.msdn.com>. The Windows Phone Marketplace procedures are covered in more depth in the appendix. Windows Phone applications cannot be distributed as stand-alone packages. In order to develop your own application, you need to enable your device to allow the deployment of XAP files.

Once your account has been verified by the AppHub, you can launch the Windows Phone Developer Registration tool from the Windows Phone Developer Tools folder in your Start Menu. This tool, shown in figure 1.3, will prompt you to enter your AppHub credentials and select a connected phone. You need to plug the device to your PC and pair it to the Zune client and have your PC connected to Internet in order to connect to Microsoft's registration servers.

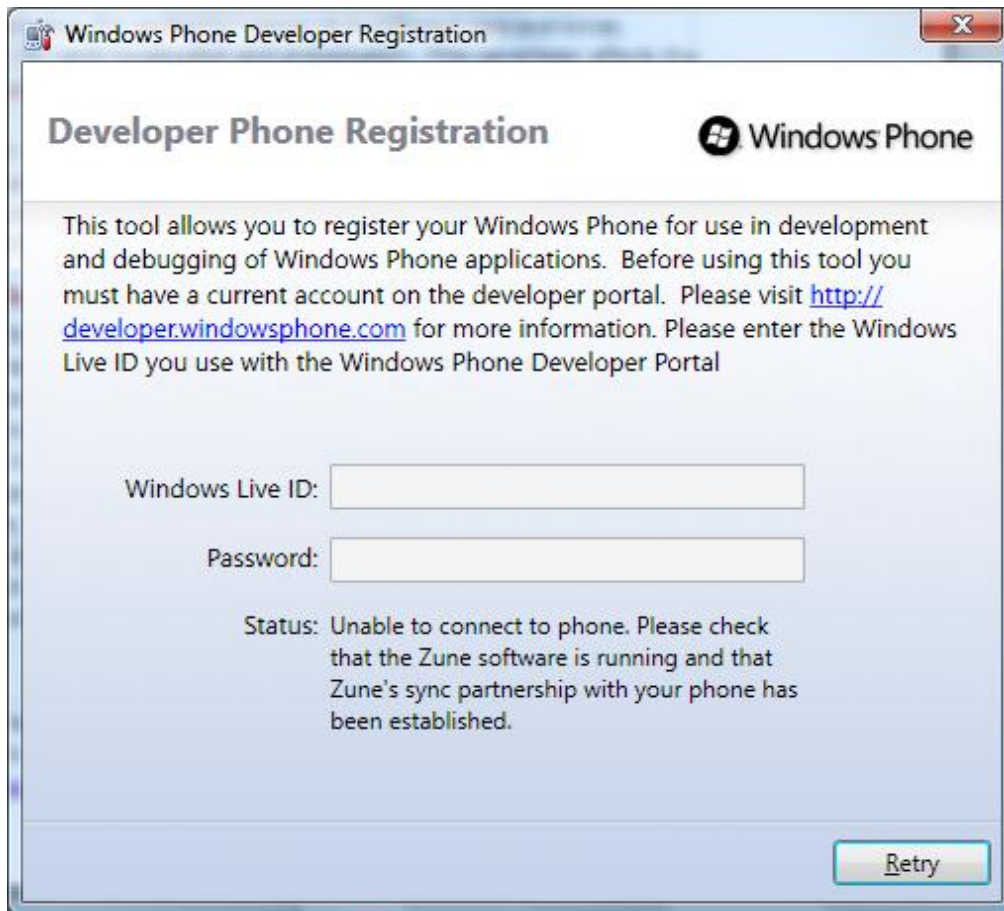


Figure 1.3 Windows Phone Developer Registration tool

There are limits to the number of phones that can be registered to a single account. There are also limits to the number of developer applications that can be installed on a phone at the same time. If you reach the installed application limit, you must uninstall one or more developer applications before you will be able to deploy a new application from Visual Studio. Occasionally your phone registration will expire and you will receive an error when attempting to deploy an application to a device, and you will need to reregister the phone with the registration tool.

The Windows Phone Developer Registration tool can also be used to unregister a phone. If you need to unregister a phone, but do not have it available because it has been lost or broken, you can unregister the device from your AppHub account's profile page, accessed through your browser at <https://users.create.msdn.com/Account/Profile>.

### 1.4.6 XAP Deployment Tool

To support testing application by non-developer team members, you can deploy just the executable binary of a Windows Phone application (the .XAP file) to the emulator or to a registered phone using the XAP Deployment tool. The XAP Deployment tool, shown in figure 1.4, is launched from the Start Menu->Windows Phone Developer Tools folder. You only need to select the target device and the XAP file, and then click the Deploy button.



Figure 1.4 Application to deploy a binary (XAP) file to the device.

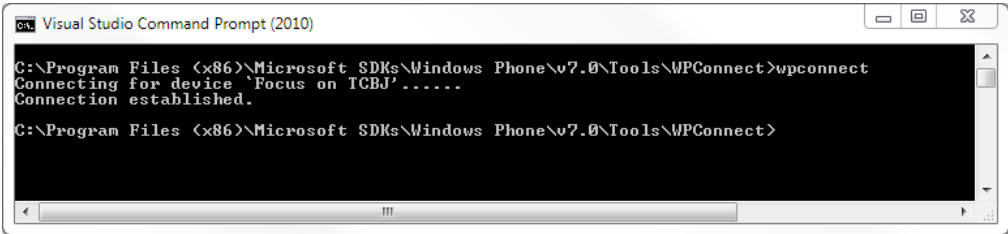
When the deployment is complete, the application can be started from the Application List. An application can be uninstalled from the Application List as well. Tap and hold the application's icon until the context menu appears and select the uninstall option.

### 1.4.7 WPCoconnect

When you use Visual Studio to debug applications running on a real phone, the phone must be connected to your computer. Is it not sufficient to have your phone connected via the USB cable; a connection must also be established via software. Usually the Zune software handles this connection.

When a phone is connected to Zune, the phone's pictures and media databases are locked. You will experience errors if you attempt to debug software that uses these libraries. Microsoft has provided the WPCoconnect tool to allowing Visual Studio to connect a phone without running the Zune software.

The WPCoconnect tool is installed by the Windows Phone Developer Tools. You can find the tool in %ProgramFiles%\Microsoft SDKs\Windows Phone\v7.1\Tools\WPCoconnect. Before running WPCoconnect, you must connect your phone and launch the Zune software. Close the Zune software once you verify that the phone was found. When you run WPCoconnect you will see a confirmation message like that shown in figure 1.5.



```

cs Visual Studio Command Prompt (2010)
C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.0\Tools\WPCoconnect>wpcconnect
Connecting for device 'Focus on ICBJ'.....
Connection established.
C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.0\Tools\WPCoconnect>

```

Figure 1.5 The WPCoconnect confirmation message

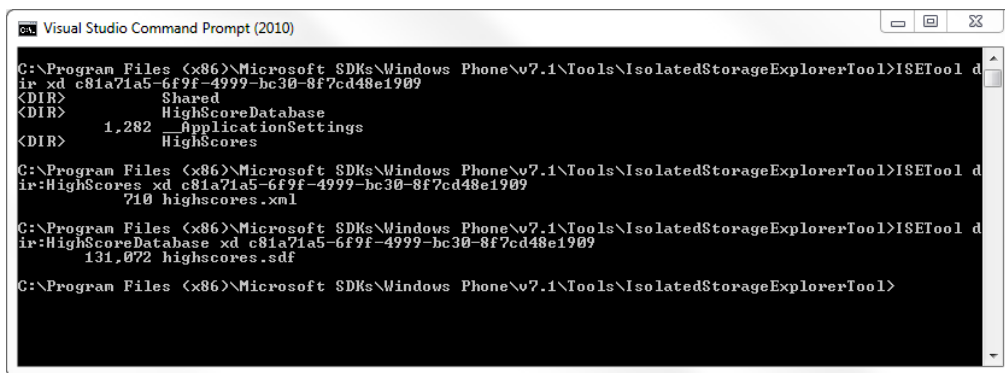
Of course, you can use this tool when you are debugging any application, not just applications that use the media library.

### 1.4.8 Isolated Storage Explorer Tool

Most applications require some form of data storage – from user preferences and user created data to local caches of data stored in a cloud application or web service. Each application is allotted their own storage sandbox on the phone, isolated from all other applications and from the operating system. Isolated storage will be empty when an application is first deployed to the emulator or a device. During execution, many applications will store data and settings in isolated storage.

While testing and debugging an application, developers might want to examine the files written to isolated storage or maybe even write data files to isolated storage to facilitate

testing. The Isolated Storage Explorer Tool (ISETool) is included in the Windows Phone 7.1 SDK to enable these scenarios. The ISETool allows a developer to take a snapshot of an application's isolated storage, copying the files from the phone to a desktop folder. The ISETool can also be used to copy files from the desktop to an application's isolated storage folder. The ISETool will also list the files in an isolated storage folder, as shown in figure 1.6.



```

Visual Studio Command Prompt (2010)
C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Tools\IsolatedStorageExplorerTool>ISETool d
ir:xd c81a71a5-6f9f-4999-bc30-8f7cd48e1909
<DIR>      Shared
<DIR>      HighScoreDatabase
<DIR>      1.282 ApplicationSettings
<DIR>      HighScores
C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Tools\IsolatedStorageExplorerTool>ISETool d
ir:HighScores xd c81a71a5-6f9f-4999-bc30-8f7cd48e1909
716 highscores.xml
C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Tools\IsolatedStorageExplorerTool>ISETool d
ir:HighScoreDatabase xd c81a71a5-6f9f-4999-bc30-8f7cd48e1909
131.072 highscores.sdf
C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Tools\IsolatedStorageExplorerTool>

```

Figure 1.6 Using the Isolated Storage Explorer Tool to list the files in isolated storage in the emulator.

The ISETool requires the application's Product GUID. The Product GUID is generated by the Visual Studio project templates and is declared in a project's application manifest file which is named WMAppManifest.xml. You will learn more about the application manifest in the next chapter.

We show you how to use the Isolated Storage Explorer to populate a read only database in chapter five.

### 1.4.9 Marketplace Test Kit

Once an application has been developed, it must go through an approval process run by Microsoft before being published to the Windows Phone Marketplace. This will ensure that the application conforms to Microsoft requirements for a Windows Phone application. Microsoft's requirements are detailed in the Windows Phone Application Certification Requirements available from <http://create.msdn.com>.

The Marketplace Test Kit includes a series of automated, monitored and manual tests you can use to ensure your application meets the Application Certification Requirements for Windows Phone. The Marketplace Test Kit also helps you assemble the graphics and screen shots that will be submitted along with the application's .xap file. The Marketplace Test Kit is installed when you install the Windows Phone Developer Tools. A screen shot of the Marketplace Test Kit is shown in figure 1.7. The Marketplace Test Kit is an extension to Visual Studio and is accessed from the *Open Marketplace Test Kit* option in Visual Studio's *Project* menu.

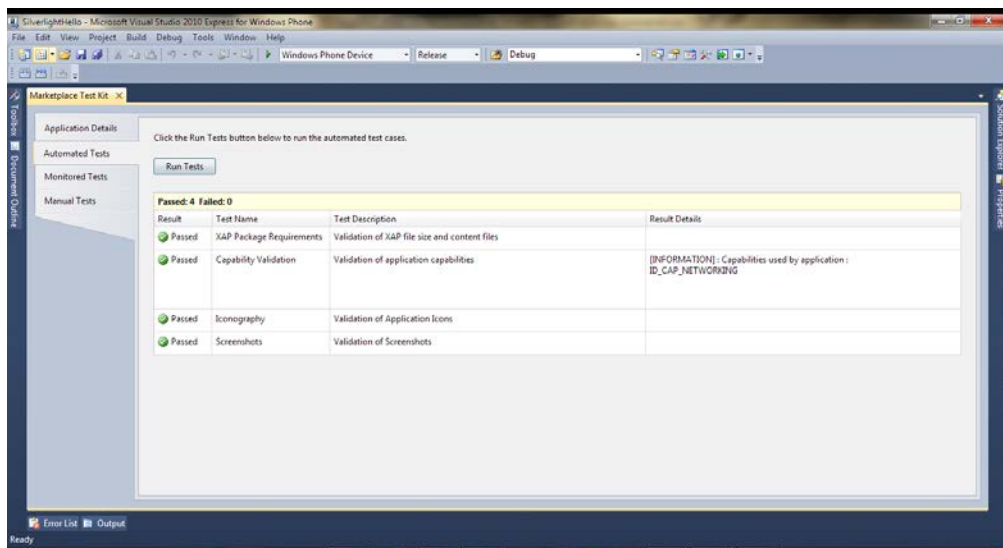


Figure 1.7 The results of automated tests performed by the Marketplace Test Kit, including a list of security capabilities used by the application.

If your application violates any of the tests, it will be rejected during the submission process. Nobody likes to receive rejection letters. While you should run the ingestion tool before you submit your application, you may also choose to run it periodically during development so that you can detect and fix issues as early as possible.

The Capability Validation test reports the security capabilities required by an application. Windows Phone is all about security sandboxes and user disclosure. Security capabilities are one face of the operating system's security model and a Windows Phone application must declare which capabilities, or features of the operating system it uses. The capabilities used by an application are declared in the application's manifest file. When an application is submitted to the marketplace, the certification process inspects the compiled code and updates the manifest with the discovered capabilities. Table 1.1 details the set of capabilities that can be listed in the manifest.

Table 1.1 Security Capabilities

Capability ID	Description	Required by
ID_CAP_APPOINTMENTS	Access appointment data from the calendar	Microsoft.Phone.UserData.Appointments
ID_CAP_CONTACTS	Access contact data from the address book	Microsoft.Phone.UserData.Contacts

ID_CAP_GAMERSERVICES	Use Xbox Live APIs	Microsoft.Xna.Framework.GamerServices
ID_CAP_IDENTITY_DEVICE	Access to device information	Microsoft.Phone.Info.DeviceExtendedProperties
ID_CAP_IDENTITY_USER	Access to user information	Microsoft.Phone.Info.UserExtendedProperties
ID_CAP_ISV_CAMERA	Access the Camera API and the raw image stream	Microsoft.Devices.PhotoCamera
ID_CAP_LOCATION	Use location services	System.Device.Location
ID_CAP_MEDIALIB	Access the media library	Microsoft.Devices.MediaHistory Microsoft.Devices.Radio.FMRadio Microsoft.Xna.Framework.GamerServices Microsoft.Xna.Framework.Media System.Windows.Media.MediaStreamSource
ID_CAP_MICROPHONE	Record with the microphone	Microsoft.Xna.Framework.Audio.Microphone
ID_CAP_NETWORKING	Use network services	Microsoft.Phone.Controls.WebBrowser Microsoft.Phone.Notification Microsoft.Xna.Framework.GamerServices System.Net
ID_CAP_PHONEDIALER	Initiate phone calls	Microsoft.Phone.Tasks.PhoneCallTask
ID_CAP_PUSH_NOTIFICATION	Receive push notifications	Microsoft.Phone.Notification

ID_CAP_SENSORS	Use the Accelerometer	Microsoft.Devices.Sensors
ID_CAP_WEBBROWSERCOMPONENT	Use the web browser control	Microsoft.Phone.Controls.WebBrowser
ID_HW_FRONTCAMERA	Access the forward facing camera	Microsoft.Devices.PhotoCamera

The manifest file created by the Visual Studio project templates automatically declares every capability, except ID\_HW\_FRONTCAMERA. The developer can remove any of the capabilities that are not required for their application. During marketplace certification, the list provided by the developer is deleted and replaced by a list of capabilities detected by the certification tools. The assembly is examined for calls to the secured APIs and when one is found, the matching capability is re-added to the manifest. If required capability is not listed in the manifest, the secured API will throw a `UnauthorizedAccessException`. When an application is downloaded from the marketplace, the list of capabilities used by an application is displayed to the user, allowing the user to make an informed decision before purchasing the application.

## 1.5 Summary

This chapter has been an introduction to the Windows Phone platform. Windows Phone 7 is not an upgrade of Windows Mobile, but is an entirely new operating system. Developers moving to Windows Phone from Windows Mobile, or desktop application, must learn to work with the Windows Phone Developer Tools. Windows Phone 7 is locked down pretty tight, and many types of applications simply cannot be ported to Windows Phone 7.

You'll see in the next chapter how easy it is to create Windows Phone application. Hopefully the ease of development mitigates the lack of advanced functionality that many developers have come to expect from Windows based platforms.

Now, install the development tools and move to the next chapter: it's time to code!