**Windows Mobile Application Development**

**(Part 2)**

**Topic 1: Device Emulator and Device Emulator Manager**



[**http://msdn.microsoft.com/rampup**](http://msdn.microsoft.com/rampup)

**Introduction**

If you are a Windows Mobile Application Developer you probably have heard or have already used the Device Emulator. This is a priceless tool to test your applications on a variety of different Windows Mobile devices with all kinds of different form factors, without the need to have access to a large range of physical devices.

This article provides information to use Device Emulator to test your applications, how to configure Device Emulator to control the behavior of the tool. This article also provides information on how to make use of Device Emulator in combination with Cellular Emulator to allow you to test phone calls and sending and receiving of text messages. Finally, this article also contains a section on how to make use of Device Emulator Manager programmatically to control different emulators from inside a desktop application.

**Different Versions of Device Emulator**

Device Emulator 1.0 shipped with Visual Studio 2005. This emulator allows you to run and test your applications without the need to have access to a real Windows Mobile device. Since Device Emulator is a true ARM emulator, it will run exactly the same executables that also run on real devices. Building upon Device Emulator 1.0 but extending its functionality and performance, Device Emulator 2.0 shipped with the Windows Mobile 6 SDKs. As part of the Windows Mobile 6 SDK installation process, Device Emulator 2.0 automatically replaces Device Emulator 1.0. This means that all of your Windows Mobile Smart Device project that you developed with Visual Studio 2005 will use Device Emulator 2.0 once it has been installed. Even if you are not targeting Windows Mobile 6 devices and are still using Visual Studio 2005 for device development, it makes sense to install at least one of the Windows Mobile 6 SDKs to take advantage of Device Emulator 2.0. Not only will your code execute faster on Device Emulator 2.0, it also gives you lots of new features like extended configuration options to emulate particular hardware settings and the possibility to test wireless communication connections.

Device Emulator 3.0 is the latest version of Device Emulator. It ships with Visual Studio 2008, and again automatically replaces any previous version of Device Emulator that was installed on your development machine as part of the Visual Studio 2008 installation process. Device Emulator 3.0 offers the same performance improvements as Device Emulator 2.0. Using Device Emulator Manager’s exposed COM Interfaces it is also possible to control emulators programmatically, for instance to automate the process of starting and cradling emulators or to programmatically alter emulator configurations.

**Using Device Emulator inside Visual Studio 2008**

There are a number of ways to make use of Device Emulator from within Visual Studio 2008. To start working with Device Emulator, you can simply select one of your installed emulator images that match the SDK for which your application was build as Target Platform inside Visual Studio 2008 and just start debugging your application. Prior to running your application on the selected Device Emulator, the application will be build, the Device Emulator will be started if it was not already running and it will connect to Visual Studio 2008. After the application is deployed, optionally together with the right version of the .NET Compact Framework as well as with the SQL Server CE runtime, it will start automatically on Device Emulator.

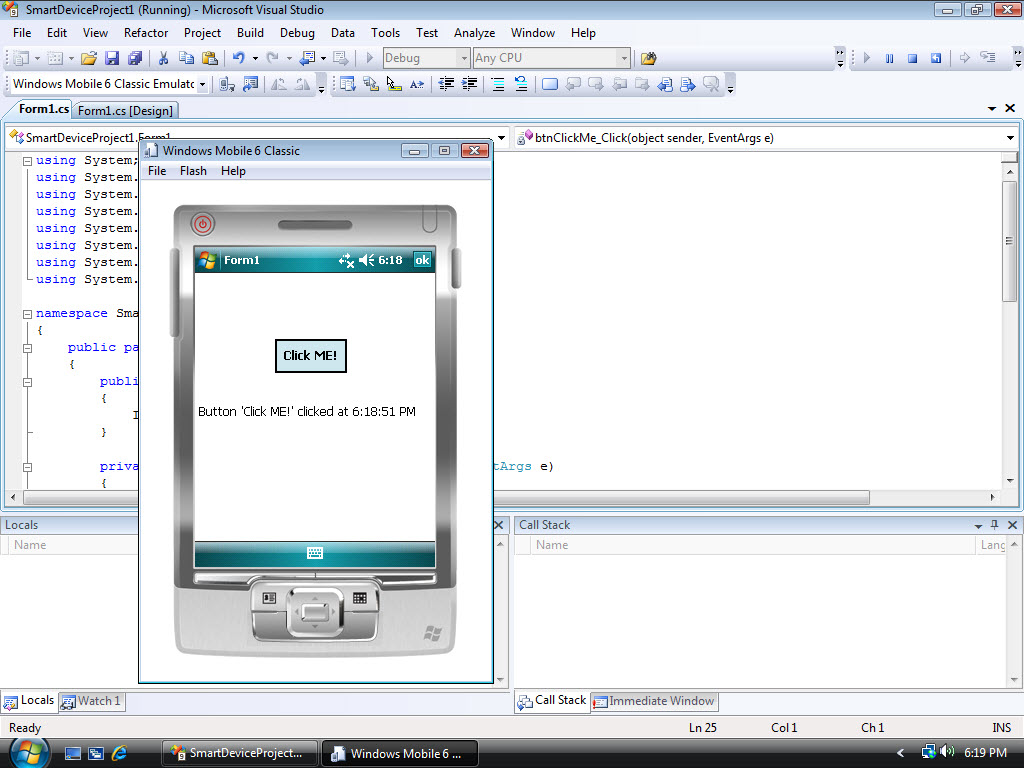


Figure - Windows Mobile application running in Device Emulator

**Setting Device Emulator Properties**

In many situations you probably want to use different hardware settings when testing your Windows Mobile application. For those scenarios Device Emulator allows you to emulate a number of different peripherals. For instance, you can specify a particular battery charge level to test your application under low battery conditions in a very easy way. You can also bind a virtual network adapter to a physical network adapter that is available on your development machine and you can map virtual COM ports of Device Emulator to physical COM ports. Another option you have is to share a folder between your development machine and Device Emulator using the Device Emulator Properties. The shared folder will appear as a Storage Card inside Device Emulator. Making use of folder sharing allows you to transfer files back and forth between Device Emulator and your development machine.

You can display the Emulator Properties dialog box from inside Device Emulator by clicking **Configure** on the Device Emulator **File menu**.

You can display the same dialog from inside Visual Studio 2008 by clicking **Options** on the Visual Studio 2008 **Tools menu**, expanding the **Device Tools** in the **Options dialog box** and then click on **Devices**. In the **Devices box** you can then select an emulator and click on **Properties**. The **Emulator Properties dialog box** can now be activated by clicking on **Emulator Options**.

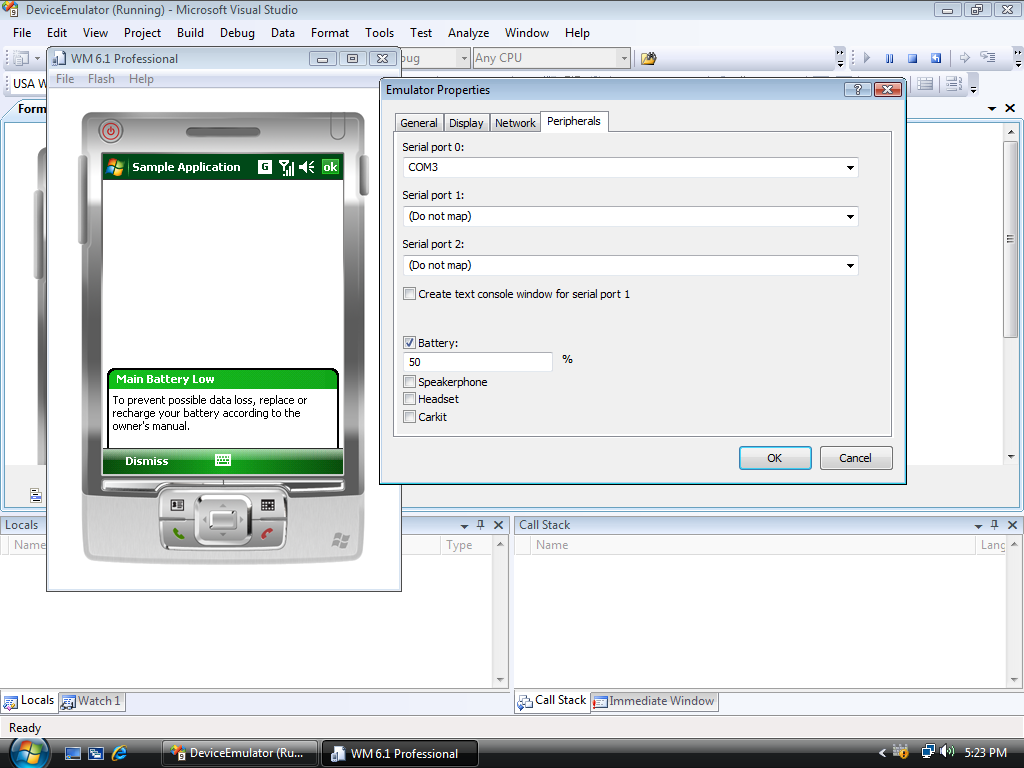


Figure - Configuring Emulator Properties

**Using Device Emulator in combination with Cellular Emulator**

There are many situations where you might want to test cellular connectivity in combination with your own applications. For instance, your application might have functionality to make phone calls, or maybe your application can intercept SMS messages to take some action upon. In scenarios like the one described, Device Emulator is very useful, because it allows you to test cellular connectivity without the need to have a physical cellular connection. Working together with Device Emulator, Cellular Emulator can be used to emulate cellular connections. Cellular Emulator ships with Device Emulator 2.0 and higher. You can start Cellular Emulator by navigating to  
**Start**/**All Programs** / **Windows Mobile 6 SDK** / **Tools** / **Cellular Emulator** in the **Start menu**.

Before you can use Device Emulator in combination with Cellular Emulator you have to map Serial port 0 of Device Emulator to the serial port that is shown in the status bar of Cellular Emulator. In figure 3 you can see how **Serial port 0** is mapped in the Emulator Properties screen to **COM3** to establish a connection to Cellular Emulator. After mapping the serial port you have to reset Device Emulator before you can make use of Cellular Emulator’s functionality.

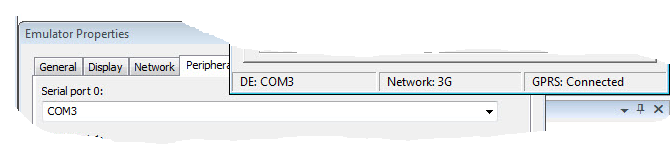


Figure - Connecting Cellular Emulator to Device Emulator

Amongst others you can

* Make phone calls from Device Emulator
* Receive phone calls on Device Emulator
* Use a GPRS connection inside Device Emulator to call out to a network
* Send and receive SMS or text messages
* Select between 2G and 3G networks
* Analyze low level modem commands

**Setting up a connection through Cellular Emulator**

As soon as Device Emulator and Cellular Emulator are connected you can use Cellular Emulator to test a lot of functionality regarding cellular connectivity. You can immediately make phone calls to and from Device Emulator and send SMS messages to and from Device Emulator. You can also setup a data connection to connect to a network through GPRS. In order to do so, you have to setup and configure the data connection on Device Emulator. More information about setting up Device Emulator to make use of a data connection can be found here: <http://msdn.microsoft.com/en-us/library/bb158505.aspx>.

In figure 4 you can see several data connections being active between Device Emulator and Cellular Emulator. In the background on Device Emulator you see a running instance of Internet Explorer that displays a particular website. The information is retrieved through a GPRS connection, as is shown in the status bar of Cellular Emulator, where you can see that GPRS is connected. Cellular Emulator simply emulates a GPRS connection by making use of network connection of the host machine.

In figure 4 you can also see that Cellular Emulator’s Call Manager is used to make a phone call to Device Emulator. In the **Phone Number** edit box you can simply specify a phone number and click on the **Dial** button. Immediately you will see a notification inside Device Emulator with an incoming phone call. You can see that the phone number inside Device Emulator corresponds with the phone number that was dialed from inside Cellular Emulator.

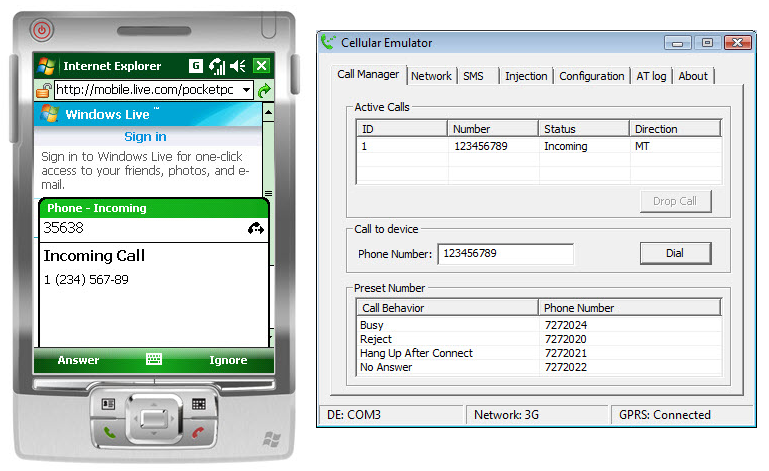


Figure - Device Emulator and Cellular Emulator in action

**Sending and receiving SMS Messages**

An important feature of Cellular Emulator is the possibility to send SMS messages to Device Emulator and to receive SMS messages from Device Emulator. Simply receiving SMS messages on Device Emulator is not that exciting. Once an SMS message is received, the user will be notified and the SMS message will be stored in the Inbox inside Device Emulator. However, you can also capture SMS messages inside your own application. This allows for lots of interesting scenarios where you can control an application that is running on a Windows Mobile device remotely. It is up to the application to either intercept SMS messages and use them exclusively or to intercept SMS messages and afterwards send them on to the Inbox on the device.

Especially in those situations where you are intercepting SMS messages to control your own application you probably want to do a lot of testing, meaning you need to send many different SMS messages to your device. The combination of Device Emulator and Cellular Emulator is priceless for these kinds of test scenarios, since you don’t need a physical network or a costly plan with a telecom operator to test the entire application. Since this article is limited to explaining Device Emulator itself we will not go into details on how to intercept SMS messages programatically. If you want to know more about this topic you can take a look at the [MSDN documentation](http://msdn.microsoft.com/en-us/library/microsoft.windowsmobile.pocketoutlook.messageinterception.messageinterceptor.aspx).

In Figure 5 you can see an example of an application that captures SMS messages that start with the string “Control Message”. Each time SMS message are received on Device Emulator with a sample application called **SMS Capture** running, these messages are intercepted by the application. The application shows the body text of the SMS message in a text box and also sets the Text property of a LinkLabel control to the phone number of the sender of the SMS message, allowing the user to make a phone call to the sender of the message by simply clicking on the displayed phone number.

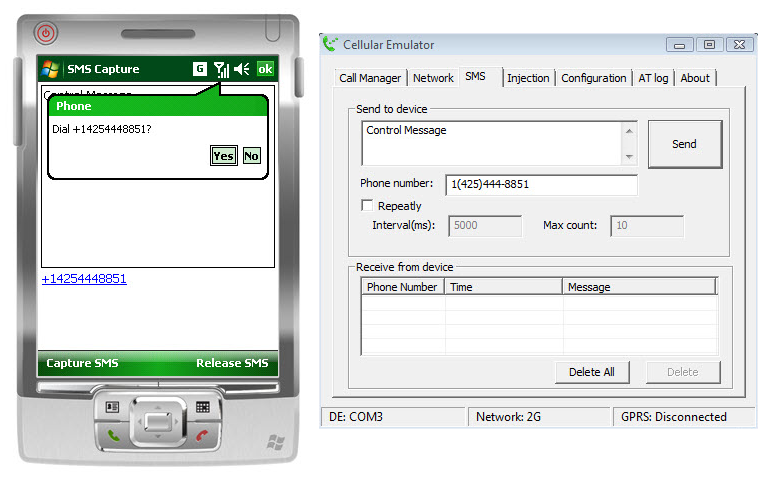


Figure - Capturing SMS Messages inside an application

**Controlling Device Emulator with Device Emulator Manager**

Visual Studio 2008 contains a tool called Device Emulator Manager that can be used to control all Device Emulators that are installed on a development machine. You can start Device Emulator Manager from inside Visual Studio by selecting **Device Emulator Manager** from the **Tools** menu. Device Emulator Manager displays a list of all installed Device Emulators in a tree view and allows you amongst others to connect to a Device Emulator, to cradle it, to uncradle it and to shutdown a Device Emulator. The ability to cradle Device Emulator is very useful, because it allows Device Emulator to connect to a development machine through ActiveSync (on Windows XP) or through the Windows Mobile Device Center (on Windows Vista). Once cradled, it allows Device Emulator to synchronize content, to call out to the Internet, to install applications on Device Emulator from the development machine using a CAB file and so on.

NOTE: You must have ActiveSync or the Windows Mobile Device Center installed to be able to cradle Device Emulator. You must also make sure that you allow connections through **DMA** (Direct Memory Access). You can take a look at the [MSDN Documentation](http://msdn.microsoft.com/en-us/library/aa188173.aspx) to learn more about cradling Device Emulator.

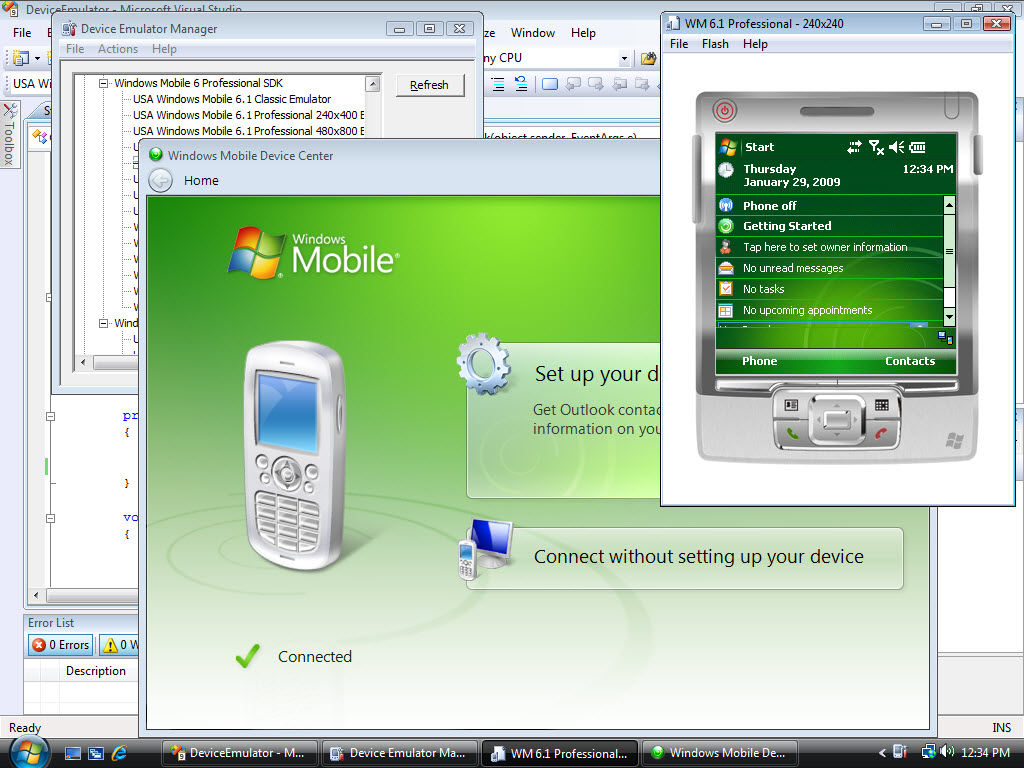


Figure - Device Emulator Manger - Cradling a Device Emulator

In Figure 6 you can see a Windows Mobile 6.1 Professional Emulator with a square screen that was started using Device Emulator Manager. After starting the emulator, Device Emulator Manager was also used to cradle it. Controlling emulators from inside Device Emulator Manager can be done by either **right clicking** on one of the emulators that are displayed in the tree view that shows all installed emulators, or by selecting an entry from the **Actions** menu on an already running emulator. Once you cradle Device Emulator you will see either ActiveSync or the Windows Mobile Device Center becoming active to establish a partnership with Device Emulator.

**Saving Device Emulator Configurations for later use**

With Device Emulator Manager it is possible to save your own particular configurations for any type of Device Emulator. Using your own configurations, you can for instance create an additional Device Emulator that always automatically start with a battery level of 50% and with a storage card configured which points to a particular folder on your development machine. To begin creating your own emulator configuration, select one of the already installed emulators from inside Device Emulator Manager, **right-click** on it and select **Connect**. Once the emulator is running you can change its configuration, after which you can save this particular configuration from inside Device Emulator Manager by again **right-clicking** it and select **Save As**. The newly created emulator configuration appears as a brand new emulator inside Device Emulator Manager. It is displayed in the tree view of Device Emulator Manager under the **My Device Emulators** node. Creating a new emulator configuration only means that a new device emulator configuration file is created and stored, since the newly created emulator configuration is already based upon a particular emulator image. Therefore, creating new emulator configurations does not take much storage space on your hard drive.

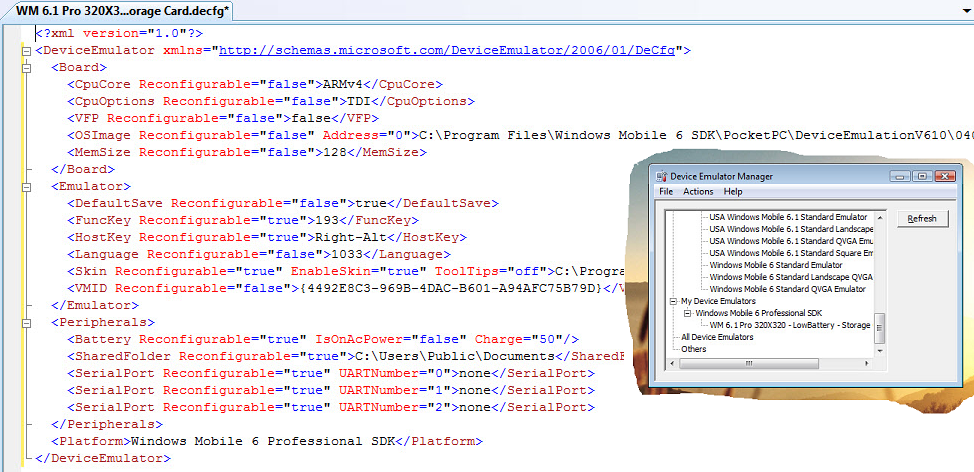


Figure - Device Emulator Configurations

In figure 7 you see an example of a device emulator configuration with a storage card defined that points to C:\Users\Public\Documents on the development machine and that emulates running on battery power with the battery charged for 50%. Each time this particular Device Emulator is started, it has these initial settings.

**Using Device Emulator to test different security configurations**

When you are using Device Emulator to test your Windows Mobile applications, you will typically not have any difficulties deploying and running them, assuming you have a working connection between Visual Studio 2008 and Device Emulator. Device Emulator acts like a perfect Windows Mobile device with no security enabled. Depending on the security settings on the physical Windows Mobile target devices, you might have restricted access to the device. Using Visual Studio 2008, you can configure Device Emulator to work in a particular security configuration. A description of different security configurations can be implemented for Windows Mobile devices is beyond the scope of this article. If you want to find more information about device security configurations you can take a look at the [MSDN documentation](http://msdn.microsoft.com/en-us/library/bb416353.aspx). Setting a particular security configuration on the Device Emulator is fairly simple and straight forward.

Once you have started a particular Device Emulator, you can retrieve its current security setting by going to the **Tools** menu of Visual Studio 2008 and selecting **Device Security Manager**. This results in displaying the Device Security Manager inside the working area of Visual Studio 2008. With Device Security Manager you can view the current security settings of the running Device Emulator and you can also view installed certificates.

NOTE: You can also use Device Security Manager to interrogate physical Windows Mobile devices.

To deploy a new security configuration to Device Emulator, you should take the following steps.

1. Select the **Security Configuration** tab inside **Device Security Manager**
2. Select a particular configuration from the list titled **Security configuration**
3. Deploy the selected configuration to Device Emulator by clicking the **Deploy to Device** tab inside Device Security Manager

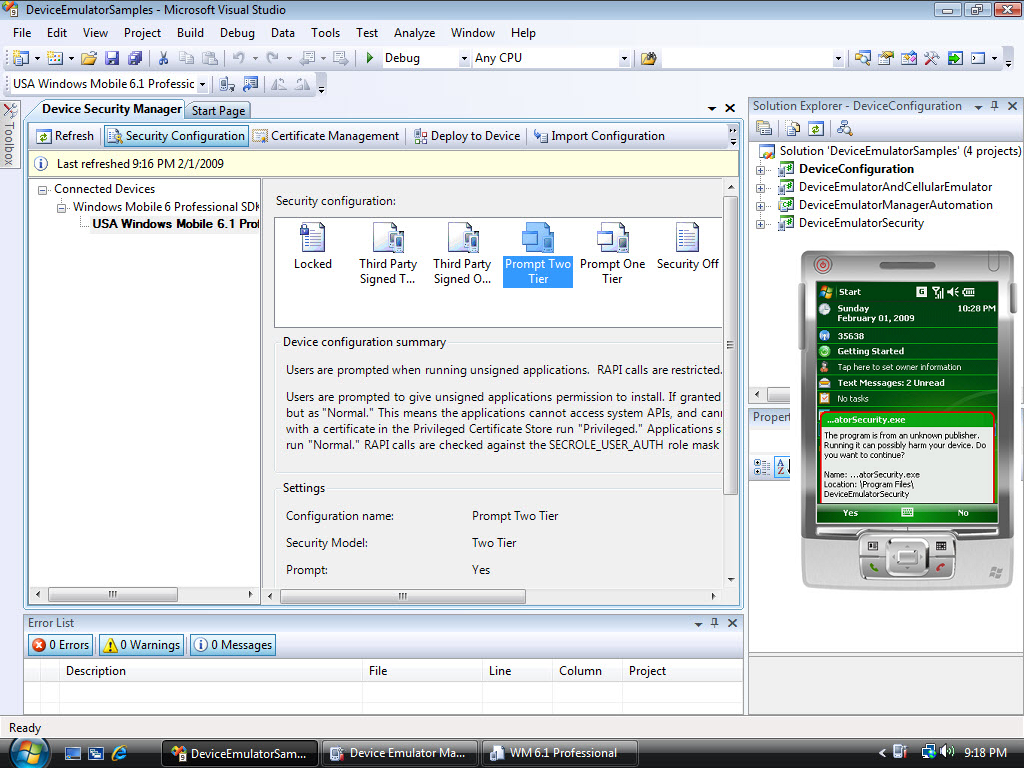


Figure - Device Security Manager in combination with Device Emulator

# **Automating Device Emulator Manager**

The Device Emulator Manager functionality is exposed as a series of COM objects that can be used to create a desktop application to control Device Emulator Manager programmatically. This means that you can for instance write a desktop application in C# that can start and cradle a particular Device Emulator. Take the following steps to create a desktop application that makes use of Device Emulator Functionality:

1. Use Visual Studio 2008 to create a Visual C# Windows application
2. Add a reference to the **Microsoft.DeviceEmulatorManager.Interop.9.0** assembly  
     
   NOTE: Even though you might be running a later version of Device Emulator, the needed assembly will still be found in the   
   \Program Files\Microsoft Device Emulator\1.0\Microsoft.DeviceEmulatorManager.Interop.9.0.dll   
   folder because each new version of Device Emulator is an in-place upgrade of a previous version of Device Emulator
3. To add code to display all data stores containing emulator images on your development machine in a tree view control, you can for instance use this code snippet in your main form’s Load event handler to interrogate the [**IDeviceEmulatorManager**](http://msdn.microsoft.com/en-us/library/bb531169.aspx) interface:  
     
    string categoryName;

TreeNode categoryNode;

IDeviceEmulatorManager emulatorManager = new DeviceEmulatorManagerClass();

emulatorManager.Reset();

try

{

while (true)

{

categoryName = emulatorManager.get\_Name();

categoryNode = new TreeNode(categoryName);

ListDeviceEmulatorSDKs(emulatorManager, categoryNode);

tvEmulators.Nodes.Add(categoryNode);

emulatorManager.MoveNext();

}

}

catch (COMException ex)

{

if (ex.ErrorCode != END\_OF\_DATA)

throw ex;

}

1. In a similar way you can retrieve all installed emulator families (represented by SDKs by creating a similar while loop, using the [**IEnumManagerSDKs**](http://msdn.microsoft.com/en-us/library/bb531190.aspx) interface
2. After retrieving all emulator families, you can in yet another similar way retrieve all particular emulator images belonging to an SDK in another while loop, using the [**IEnumVMIDs**](http://msdn.microsoft.com/en-us/library/bb531183.aspx) interface
3. Once you have access to a particular Device Emulator, you can call methods to control that emulator, for instance by calling its **Connect**, **Cradle**, **Uncradle** and **Shutdown** methods on the [**IDeviceEmulatorManagerVMID**](http://msdn.microsoft.com/en-us/library/bb531187.aspx) interface

More information on the automation possibilities of Device Emulator Manager can be found in this article.

# **Additional resources and references**

* [Window Mobile Developer Center](http://msdn.microsoft.com/en-us/windowsmobile/default.aspx)
* [Device Emulator Reference](http://msdn.microsoft.com/en-us/library/aa188178.aspx)
* [How do I Automate the Setup and Test Environment that uses the Cellular Emulator](http://msdn.microsoft.com/en-us/windowsmobile/cc507091.aspx)
* [How do I Programmatically Control the Configuration of a Running Device Emulator](http://msdn.microsoft.com/en-us/windowsmobile/bb985512.aspx)
* [How do I Automate the Setup and Test Environment that uses the Cellular Emulator](http://msdn.microsoft.com/en-us/windowsmobile/bb872392.aspx)