您的潜力，我们的动力

Microsoft
微软（中国）有限公司

使用微软消息队列构建可靠的移动应用程序

高惠杰

合作伙伴技术支持工程师

合作伙伴支持部

微软全球技术支持中心 大中华区

msdn

MSDN Webcasts

MSMQ概要

通过MSMQ传递消息是一种灵活、高效的通讯途径

- MSMQ是一种内建于Windows和Windows CE操作系统的消息传递架构
- MSMQ使用一种灵活而轻量级的编程模型
- MSMQ应用程序开发人员不需要为以下问题编写额外的代码：
  - 建立和维护网络连接
  - 选择通讯渠道
  - 为永久数据开辟缓存
- MSMQ对于“偶尔连线”的分布式应用程序而言，是一种理想的解决方案
  - 提高了应用程序的可用性和可伸缩性
  - 可以使应用程序分布到多个、甚至是不稳定的网络中
  - 维持数据的完整性

MSMQ 概要

- MSMQ使得应用程序之间可以相互通讯，而又不要求实时连接
- MSMQ支持以下功能：
  - 保证消息能够送达
  - 对消息进行路由
  - 提供一定的安全保证
  - 消息优先级
- 如果目标消息队列暂时不可用（比如网络连接不通畅） MSMQ能自动重试
- 如果远程目标消息队列不可用，MSMQ可以暂时将消息保存在本地的消息队列缓存中
- 一切都是为了保证消息100%送达!

MSMQ消息队列类型

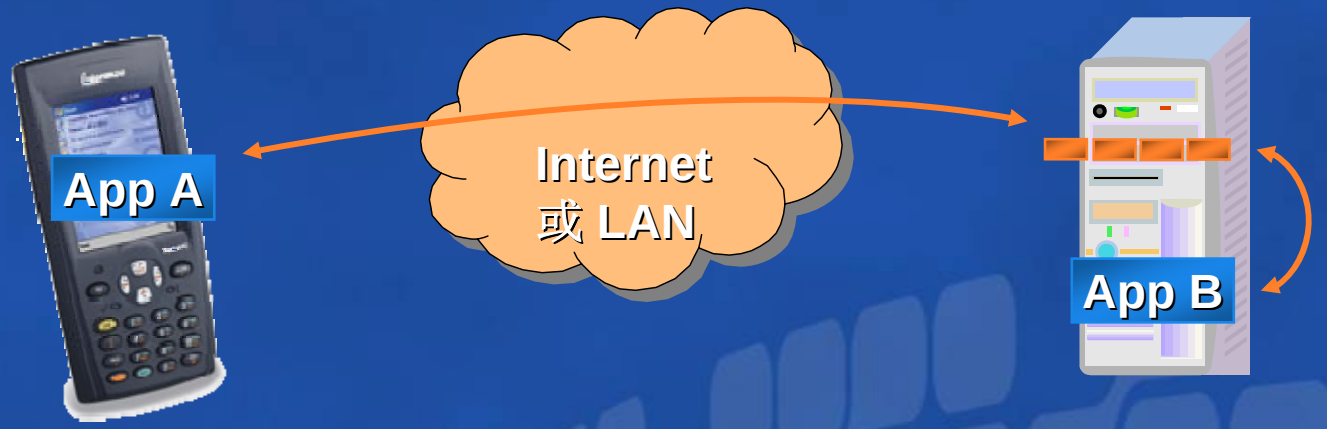
- MSMQ 支持两种不同的消息队列
  - Public
    - 消息队列必须存在于一台加入域的机器中
    - 消息队列的名称与属性通过活动目录Active Directory发布
    - Public消息队列的名称形如 *machineName\queueName*
  - Private
    - 大部分的属性和功能与public消息队列类似，但是你必须通过一些机制来找到该消息队列
    - 比public消息队列效率高，不依赖于活动目录Active Directory
    - Private消息队列的名称形如 *machineName*\Private$*queueName*
- Direct Format命名法
  - TCP:192.168.0.11\Private$\MyQueue
  - OS:MyServer\Private$\MyQueue
  - 通过使用Direct Format命名法可以提高效率

Windows CE对MSMQ的支持
关键区别

- CE上的MSMQ实现是桌面电脑上MSMQ实现的一个子集
- CE上的MSMQ相对于桌面版本主要区别在于:
  - 无法从一个远程消息队列中读取消息
  - 仅仅作为一个独立的客户端运作，无法通过查询活动目录来查找队列，故必须知道欲访问队列的详细位置，并且仅能创建本地的私有队列
  - 无法参与到MS DTC加入到分布式事务中去，但仍然支持单消息事务：保证一条消息能够有序的、不重复的被发送
- 支持NIC跟踪
  - MSMQ能够在网络连接恢复的时候尝试重新发送未发送的消息

Windows CE对MSMQ的支持

连线场景



- 设备A 发送消息至服务器B上的private或public消息队列
- 由服务器上的应用程序B 从该消息队列中接收消息
- 在Windows CE 4.0之后的版本及Windows Mobile 5.0 Pocket PC/Smartphone上,支持通过HTTP方式的Internet访问，但在Pocket PC 2003不支持

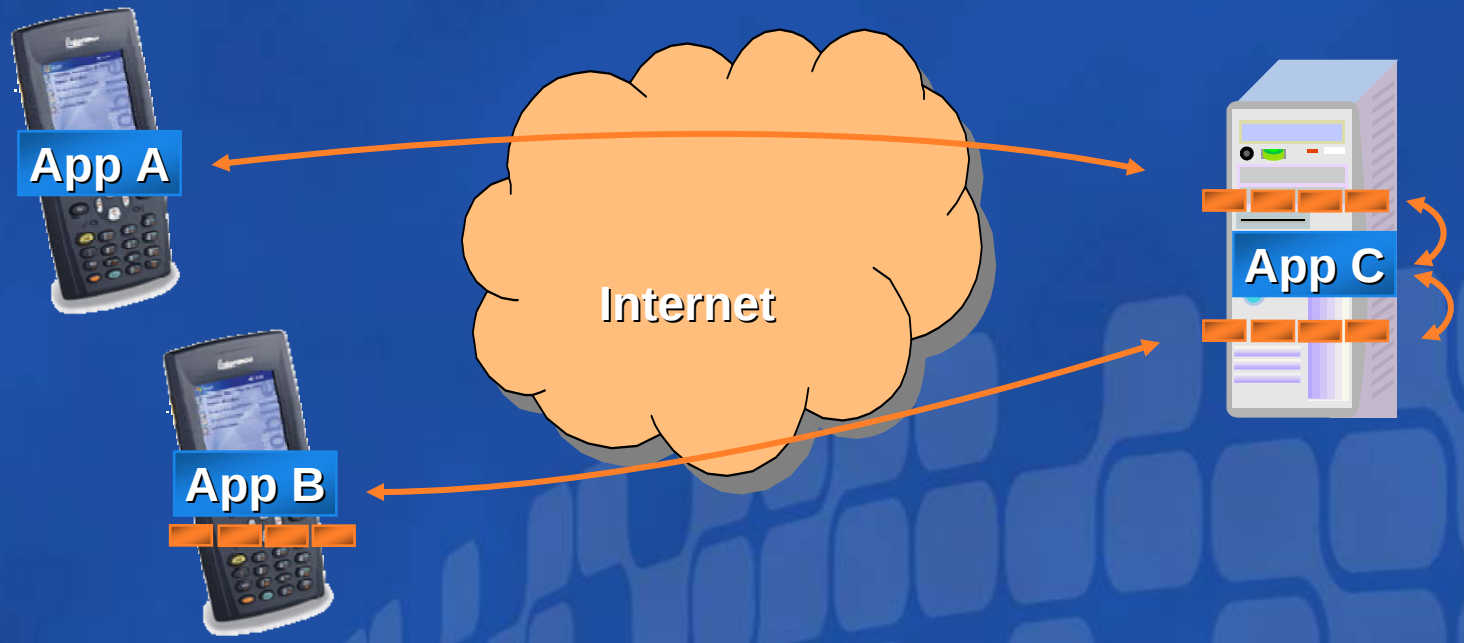
Windows CE对MSMQ的支持

连线场景

- 设备B在本地创建一个private消息队列
- 设备A向设备B上的private消息队列发送消息
- 设备B从private消息队列中接收消息
- 注意：Windows CE下的MSMQ可以向远程息消息队列写入消息，但是无法读取消息

Windows CE对MSMQ的支持

连线场景



- 设备A 发送消息至服务器C上的private或public消息队列
- 服务器C接收到消息后存储在本机上
- 设备B上的应用程序B通过间歇性地给服务器C发送消息来轮训，服务器C接收到轮训消息后会将设备A发来的消息转送给设备B上的消息队列

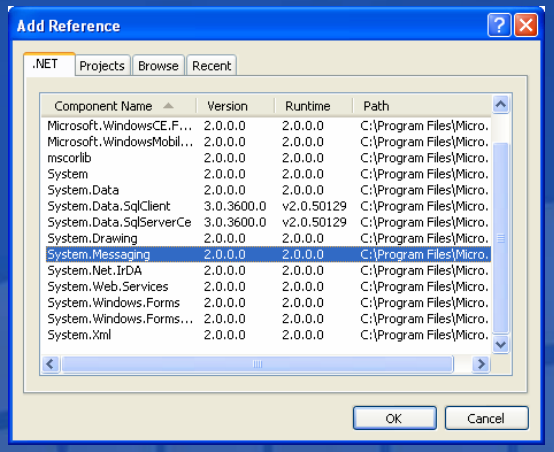
.Net CF 2.0中的MSMQ应用程序
开发环境

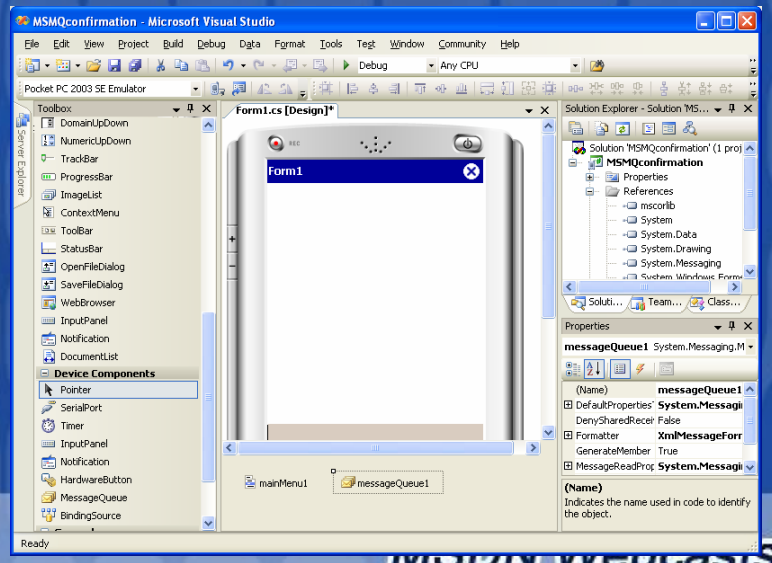
- Visual Studio 2005
- 目标设备或模拟器：
  - 针对Windows CE 4.2或Windows CE 5.0
    - 在构建过程中加载MSMQ组件
  - 针对Windows Mobile 2003 for PPC/Smartphone
    - 从SDK中安装MSMQ的组件包
  - 针对Windows Mobile 2005 for PPC/Smartphone
    - 从Windows Mobile Optional Server Components中的CAB文件安装
    - 参考http://msdn.microsoft.com/windowsmobile

.Net CF 2.0中的MSMQ应用程序

建立工程

- 通过VS向导创建一个工程
  - 在工程中添加对**System.Messaging**的引用
  - 直接将**MessageQueue**组件拖曳到设计窗口中





.Net CF 2.0中的MSMQ应用程序

发送消息


```
using System.Messaging;
...
    //Send and receive a message in button1's click event handler
    private void button1_Click(object sender, EventArgs e)
    {
        //The Queue on the device we will be sending messages to
        string strDestQ = @".\private$\queueOnDevice";
        try
        {
            // Create the queue if needed
            if (!MessageQueue.Exists(strDestQ))
                MessageQueue.Create(strDestQ);
            MessageQueue mq = new MessageQueue(strDestQ);

            // Build message
            Message m = new Message();
            m.Label = "Hello World";
            m.Body = "Hello World";

            // Send and close queue
            mq.Send(m);
            mq.Close();
            MessageBox.Show("Message sent!");
```

.Net CF 2.0中的MSMQ应用程序

接收消息


```
//continued…

            // Create an Xml formatter for a message that
            // is a String type
            mq.Formatter =
                new XmlMessageFormatter(new Type[] { typeof(String) });

            // Receive message synchronously
            Message messageReceived = mq.Receive();

            MessageBox.Show((string)messageReceived.Body);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }
```

您的潜力，我们的动力

Microsoft
微软（中国）有限公司

video

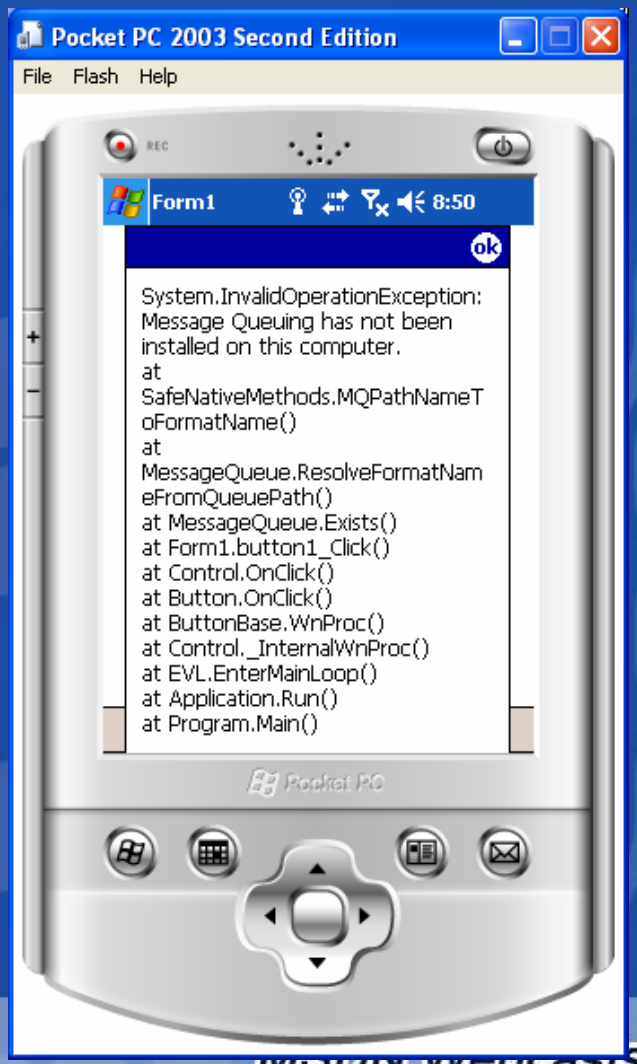
发送消息演示

msdn

MSDN Webcasts

.Net CF 2.0中的MSMQ应用程序
开始测试

- 如果应用程序抛出异常：**System.InvalidOperationException**
  - **“Message Queuing has not been installed on this computer.”**
- 这意味着您必须先在操作系统中先启动消息队列服务！



在设备上部署MSMQ
安装运行时组件

- 在基于Windows CE的设备中，如果MSMQ组件已经构建在内，则无须安装额外组件
- 在基于Windows Mobile的Pocket PC或Smartphone上，您还需要如下文件：
  - msmqadm.exe
  - msmqadmext.dll
  - msmqd.dll
  - msmqrt.dll
  - visadm.exe
- 从何处安装这些文件：
  - C:\Program Files\Windows CE Tools\wce420\{device}\Support\msmq\armv4
  - 在目录中您可以看到 \armv4 和 \x86 两个不同的平台设置，请在Visual Studio 2005下，无论是真实设备还是模拟器都是用 \armv4 版本，因为VS2005中的模拟器现在已经能直接模拟ARM的CPU了！
  - 将上述文件复制到设备中的\Windows目录
- 在基于Windows Mobile 2005的系统中：
  - 安装MSMQ CAB包

在设备上部署MSMQ
通过MSMQADM.EXE来管理MSMQ

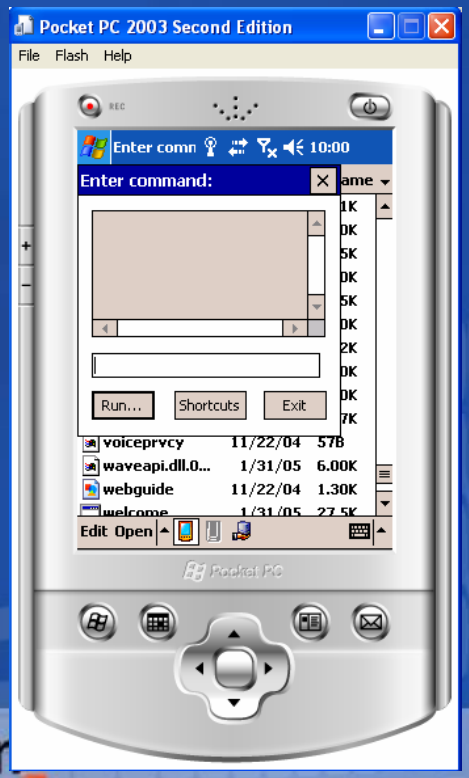
- msmqadm register install
  - 加载MSMQ驱动程序
- msmqadm register
  - 注册配置信息
- msmqadm enable binary
  - 启动消息传输协议
- msmqadm start
  - 启动MSMQ服务

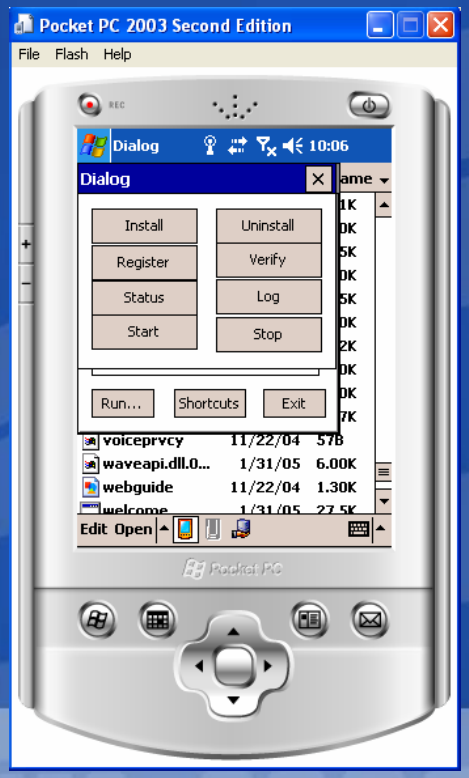
在设备上部署MSMQ
通过MSMQADM.EXE来管理MSMQ

- msmqadm status
  - 查看MSMQ状态
- msmqadm stop
  - 停止MSMQ服务
- msmqadm register cleanup
  - 清除配置信息
- msmqadm register uninstall
  - 卸载MSMQ驱动程序

在设备上部署MSMQ
通过VISADM来管理MSMQ服务

- 在文件浏览器中，运行 \windows\visadm.exe
- 选择相应的按钮来安装、注册、校验、开始、查询状态





- Install
- Register
- Verify
- Start
- Status

您的潜力，我们的动力

Microsoft
微软(中国)有限公司

video

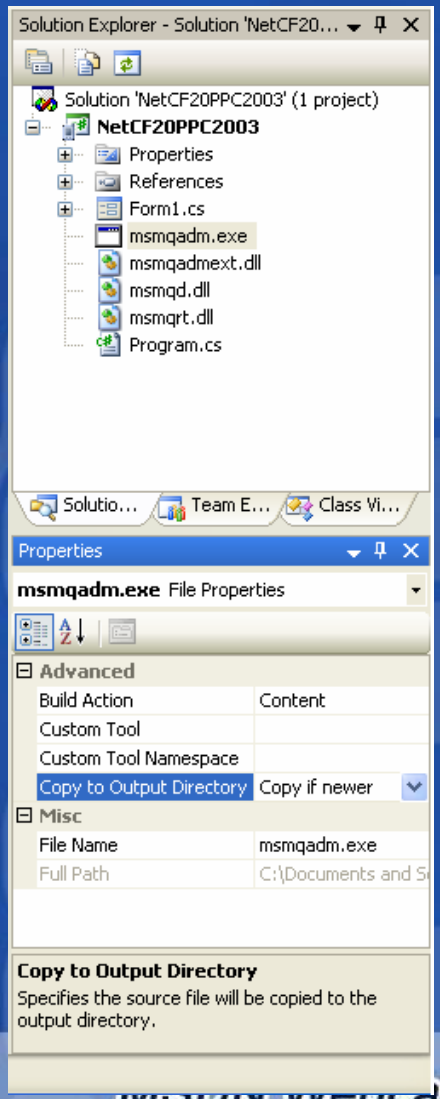
在Pocket PC 上安装和配置MSMQ

msdn

MSDN Webcasts

在设备上部署MSMQ
在程序中部署MSMQ

- 解决Visadm.exe无法自动化部署MSMQ组件的问题：
  - 通过安装工程来部署运行时组件
  - 通过程序代码来复制文件到 \Windows
  - 通过调用 CreateProcess 启动命令行工具 msmqadm.exe 来安装和启动MSMQ服务
- 如何让运行时组件随应用程序安装
  - 将 Build Action 设置为 Content
  - 将 Copy to Output Directory 设置为 Copy if Newer
- 参考：Mark Ihimoyan's Blog
  http://blogs.msdn.com/ihimmar



在设备上部署MSMQ

在程序中部署MSMQ

- 通过 System.Diagnostics.Process.Start() 启动 msmqadm.exe

```
using System.Diagnostics;
public class Utilities
    {
        // 执行 MSMQAdm.exe 命令行
        public static bool ExecuteMSMQAdm(String CmdLine)
        {
            Process prc = Process.Start(@"\windows\msmqadm.exe", CmdLine);
            if (prc == null)
                throw new ApplicationException("Process already running");

            prc.WaitForExit();
            if (prc.ExitCode != 0)
                return false;
            else
                return true;
        }

        public static void StartMSMQ()
        {
            // 检查MSMQ状态
            if (!(ExecuteMSMQAdm("status")))
            {  …
```

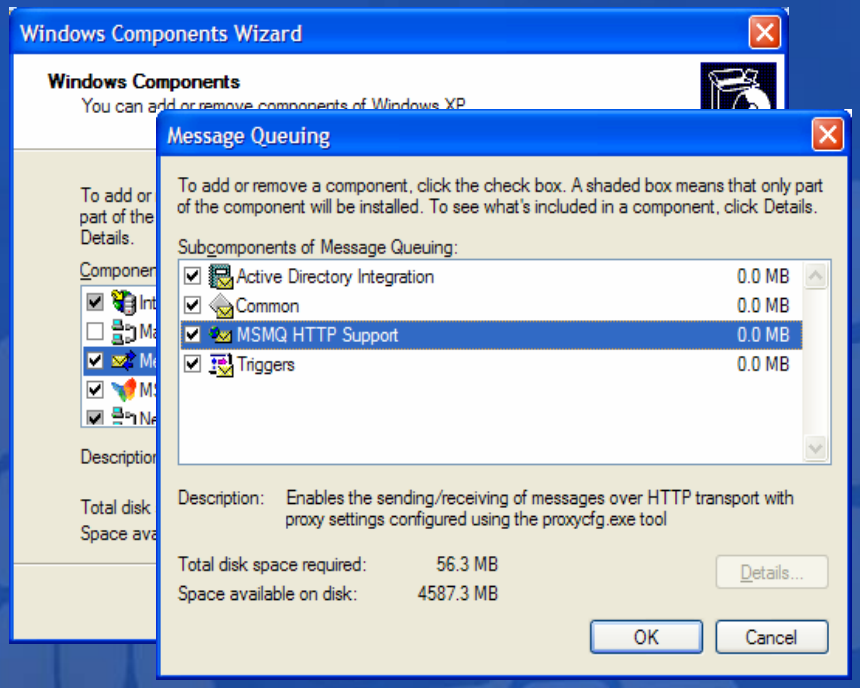
在设备上部署MSMQ
在程序中部署MSMQ

- 通过 **msmqadm.exe** 清除原有注册表信息，安装MSMQ设备驱动程序，并在注册表中创建新的注册信息的过程：

```
// 检查MSMQ状态
if (!(ExecuteMSMQAdm("status")))
{
    try {
        // 清楚原有注册信息
        ExecuteMSMQAdm("register cleanup");

        // 安装设备驱动
        ExecuteMSMQAdm("register install");

        // 注册关键信息，比如消息存放位置
        ExecuteMSMQAdm("register");

        // 打开Binary消息传递协议
        ExecuteMSMQAdm("enable binary"))
    }
    catch (Exception exc) {
        MessageBox.Show("MSMQ Install Failed! Error: " + exc.Message);
        return;
    }
    …
```

在服务器上部署MSMQ

- 默认情况下Windows XP与Windows Server 2003都没有预装MSMQ组件
  - 通过添加/删除程序中的添加Windows 组件 来安装MSMQ
  - 默认情况下，对于HTTP的支持选项没有选上

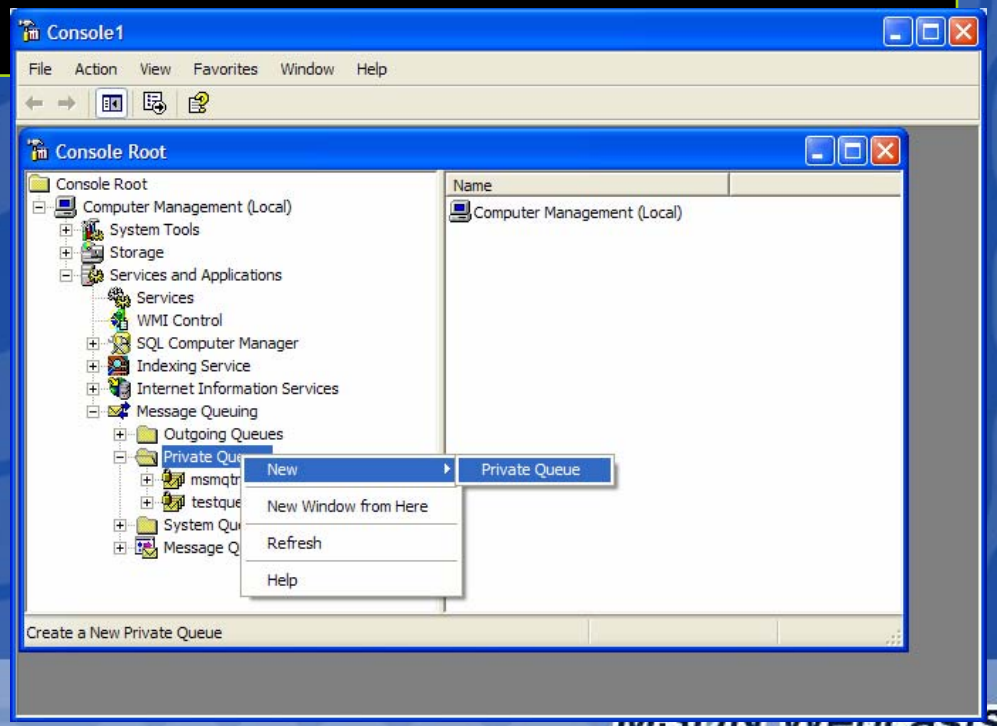


在服务器上部署MSMQ

- 通过System.Messaging中的API来创建/删除消息队列

```
try
{
   System.Messaging.MessageQueue.Create(@".\Private$\MyPrivateQueue");
}
……
catch(System.Exception)
```

- 或使用管理用户界面来创建消息队列



video

在服务器上创建一个消息队列

向远程消息队列发送消息

- 通过Direct Format命名来访问远程的消息队列
  - 通过hostname来访问远程private消息队列
    Direct=OS:*hostname*\Private$*queuename*
  - 通过IP地址来访问远程private消息队列
    Direct=TCP:*nn.nn.nn.nn*\Private$*queuename*
  - 访问远程public消息队列
    Direct={OS:*hostname*|TCP:*nn.nn.nn.nn*}*queuename*
- 特例：当需要发送一条消息到public消息队列，同时又不知道具体的机器名称时：
  - Windows CE 无法通过活动目录Active directory来定位public消息队列
  - 需要在
    HKEY_LOCAL_MACHINE\Software\Microsoft\MSMQ\SimpleClient\OutFRSQueue 中设置一个 Falcon Routing Server (FRS) 来路由消息

向远程消息队列发送消息


```
using System.Messaging;
...
    // 定义远程消息队列的路径
    string strDestQ =
        @"FormatName:Direct=OS:shaserver\private$\testqueue";
    try
    {
        // 打开消息队列
        MessageQueue mq = new MessageQueue(strDestQ);

        // Build message
        Message m = new Message();
        m.Label = "Hello World";
        m.Body = "Hello World";

        // 发送并关闭队列
        mq.Send(m);
        mq.Close();
        MessageBox.Show("Message sent!");
    }
    catch (MessageQueueException ex)
    {  …   }
```

接收消息（同步）


```
try
{
    MessageQueue mq = new MessageQueue(@".\Private$\MyQueueOnSvr“);

    // 设置格式化器，告诉MSMQ我们将会接收的消息类型
     mq.Formatter =
        new XmlMessageFormatter(new Type[] { typeof(String) });

    // 同步接收消息，设定超时为30秒
    Message messageReceived =
        mq.Receive(new TimeSpan(0, 0, 30));

}
// 捕捉异常 - 消息超时等
catch (MessageQueueException ex)
{
    MessageBox.Show(ex.ToString());
}
```

接收消息（异步）


```
Private MessageQueue asyncMQ;

public void ReceiveAsync() {
    try {
       asyncMQ = new MessageQueue(@".\Private$\MyQueueOnSvr“);

       // Create an Xml formatter for a message that
       // is a String type
       asyncMQ.Formatter =
           new XmlMessageFormatter(new Type[] { typeof(String) });

        // Add an event handler for the ReceiveCompleted event.
        asyncMQ.ReceiveCompleted +=
            new System.Messaging.ReceiveCompletedEventHandler(
                MyOwnReceiveCompleted);

        // Begin an asynchronous read operation
        asyncMQ.BeginReceive ( ) ;
    }
    catch (MessageQueueException) { … }
}
```

接收消息（异步）


```
public void MyOwnReceiveCompleted(Object source,
    System.Messaging.ReceiveCompletedEventArgs asyncResult)) {
    try {
        // 结束异步接收操作
        System.Messaging.Message msg =
           asyncMQ.EndReceive(asyncResult.AsyncResult);

       // 将消息实体转型为我们定义的类型
       String text = (String) msg.Body;

       // 显示信息
       MessageBox.Show(text, "Asynchronous Read Complete");

       // 再一次开始异步读取
       asyncMQ.BeginReceive();

    }
    catch (MessageQueueException) { … }
}
```

在MSMQ消息中传递对象

- 在桌面版本的.NET Framework 中有三个格式化器（Formatters ）：
  - ActiveXMessageFormatter
  - BinaryMessageFormatter
  - XmlMessageFormatter
- 在.NETCF 中仅有一个格式化器（Formatters ）：
  - XmlMessageFormatter
- 任何支持XML序列化的对象都可以通过MSMQ消息传递

```
MyCustomObject myObj = new MyCustomObject();
myMessageQueue.Send( myObj );
```

- 在接收对象时，必须设置 TargetTypes 或 TargetTypeNames 属性

```
XmlMessageFormatter myFormatter = new XmlMessageFormatter();
myFormatter.TargetTypes =
  new Type[] { typeof(MyCustomObject) });
myMessageQueue.Formatter = myFormatter;
Message msg = myMessageQueue.Receive();
MyCustomObject receivedObj = (MyCustomObject) msg.Body();
```

您的潜力，我们的动力

Microsoft
微软（中国）有限公司

video

开发消息队列应用程序

msdn

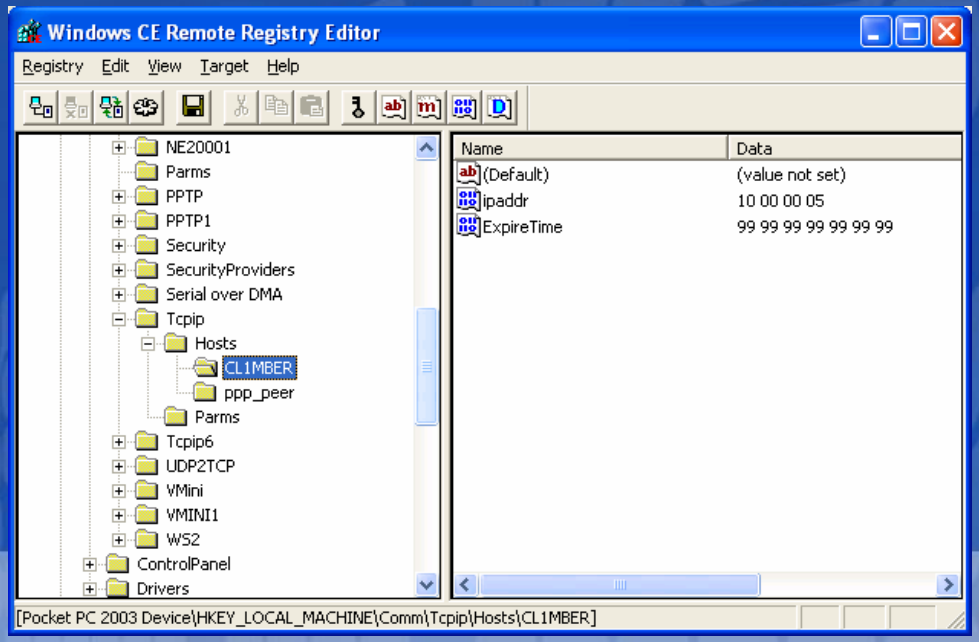
MSDN Webcasts

配置设备上的名字解析

- **Windows CE上的MSMQ对如下设置非常依赖:**
  - 解析机器名称到IP地址的能力 — DNS 或 WINS
  - 解析IP地址到机器名称的能力 — 反向DNS查询 或 Netbios查询
  - 在内部路由消息时使用机器名称而非IP地址
  - 每台设备必须使用一个唯一的设备名称
- 以下三个选项可以用来配置名字解析
  - **WINS**
  - **DNS**
  - 注册表

在注册表中配置名字解析

- 若WINS 或 DNS 不可用，则最简单的实现方式是通过配置注册表中的hosts属性
  - 在 HKEY_LOCAL_MACHINE\Comm\Tcpip\Hosts 下定义hosts
  - 内含两个二进制字段:
    - ipaddr – hex nn nn nn nn
    - ExpireTme – 99 99 99 99 99 99 99
- 图形界面配置软件: Pocket Hosts 从 http://www.handango.com 下载



自定义消息格式

- 创建一个自定义的格式化器（Formatter）
  - 创建一个实现了 IMessageFormatter接口的类
  - IMessageFormatter 仅包含三个公开的方法
    - Write
    - Read
    - CanRead
  - IMessageFormatter 也包含了 ICloneable, 故你也必须实现 Clone() 方法
  - 在自定义的格式化器（Formatter）中, 你可以：
    - 加密/解密你的消息实体
    - 实现消息压缩
  - 参考 http://support.microsoft.com/default.aspx?scid=kb;EN-US;310683

MSMQ 协议

- MSMQ 1.0, 2.0 仅支持二进制协议
- MSMQ 3.0 (Windows XP, Windows Server 2003) 支持HTTP/XML传输方式
- Windows CE 中对于 HTTP/XML 的支持：
  - Windows CE 4.0
  - 不支持Windows Mobile 2003
    - HTTP 传输需要MSXML3, 但是为了节省ROM空间Windows Mobile 2003上安装的是MSXML2
  - Windows Mobile 2005
- 设置HTTP 使用 Soap Reliable Messaging Protocol (SRMP)协议
  - 设置如下的注册表键值:
    - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\SimpleClient\SrmpEnabled 设置为 “Yes”
  - 或通过msmqadm.exe打开

```
> msmqadm enable srmp
```

通过HTTP访问MSMQ

- Direct Format 命名可以包含消息队列的URL
  - Direct=HTTP://URLAddressSpecification/msmq/private$/myQueue
  - 注意 HTTP 格式名称中可以使用斜杠(/)及反斜杠(\)，但是在URLAddressSpecification/msmq 之后才可以开始使用反斜杠(\)
  - 例如以下两种URL都是合法的：
    - Direct=HTTP://URLAddressSpecification/msmq/private$/myQueue
    - Direct=HTTP://URLAddressSpecification/msmq\private$\myQueue
- HTTP格式使得Windows CE 客户端得以方便的访问public消息队列
  - Direct=HTTP://URLAddressSpecification/msmq/myQueue
  - 通过HTTP无需Falcon Routing Server进行消息路由
- 通过HTTPS可以对消息进行加密

可靠性与性能
永久消息存储

- 保存消息的位置通过如下注册表位置来设置：
  - **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\SimpleClient\BaseDir**
  - 默认为 **\Temp\MSMQ**
- 通过将该路径指向存储卡可以使得消息即使在硬复位后仍然得以保存：
  - **\Storage Pocket PC**以及**SmartPhone**上的存储卡位置
  - 如果外部存储介质速度较慢，有可能会影响**MSMQ**的性能

可靠性与性能
可恢复消息

- **Message.Recoverable** 属性

```
// Build message
Message m = new Message();
m.Body = orderObject1;

m.Recoverable = true; // guaranteed delivery
```

  - **True**: 保证消息的送抵 (通过将消息保存在本地)
    - 慢，会消耗一定的系统资源
    - 若系统故障，消息可以被恢复
  - **False**:不保证消息的送抵 (默认)
    - 快，消耗极少的系统资源
    - 若系统故障，消息遗失

可靠性与性能
事务性消息

- 可恢复消息确认会被送抵，但是：
  - 可能被发送一次以上
  - 两条消息发送次序可能会颠倒
- 使用事务性消息可以：
  - 确保消息单次、有序的被发送
  - Windows CE MSMQ 仅支持内部、单条消息的事务，外部事务(MTS/COM+) 将不被支持
  - 创建事务性消息队列 (设备上/服务器上)
- 发送消息到事务性消息队列

```
// Create transactional message queue
MessageQueue mq = new MessageQueue(@".\Private$\TranQ", true);
```

```
Message m = new Message();
m.Body = orderObject1;
mq.Send(m, MessageQueueTransactionType.Single);
```

安全

- Windows CE 下的 MSMQ 不支持以下安全特性：
  - 用于验证的数字签名信息
  - 消息加密
  - 通过客户端证书来验证
- 如何保证HTTP事务的安全性：
  - 使用HTTPS来对通讯端口加密
  - 在Windows Server上，将来访者的IP地址做一定的限定
  - 可以使用自定义格式的消息，并在其中嵌入身份验证信息
  - 使用原生的MSMQ API在自定义SOAP请求头部
- Windows CE 下的 MSMQ 提供对拒绝服务Denial of Service (DoS) 攻击的保护
  - 对于通过HTTP发送的，基于SOAP的消息，如果其大小超过了注册表中限定的大小，则丢弃该消息
  - DefaultQuota, DefaultLocalQuota, MachineQuota

NETCF 下消息传递机制的比较

	HTTP 支持能力	自动适应/管理网络连接的能力	安全性	易开发性	服务器配置的简易性
MSMQ	✓ ✓ [1]	✓ ✓ ✓	✓ ✓	✓ ✓ ✓	✓ ✓
Web services	✓ ✓ ✓	✓ [3]	✓ ✓ ✓	✓ ✓	✓ ✓ ✓
Sockets	✓ ✓ [2]	✓ [3]	✓	✓	✓ ✓ ✓
SQL Mobile RDA	✓ ✓ ✓	✓ [3]	✓ ✓ ✓	✓ ✓	✓

Notes:

1. MSMQ在Windows Mobile 3.0或以下的版本中不支持HTTP
2. System.Net.HttpWebRequest\WebResponse直接使用Socket
3. Web services, sockets and SQL Mobile 都需要自行编码完成网络连接性的判断和管理

总结

- MSMQ CE的概念与特点
- 如何在设备上部署安装MSMQ
- 如何在本地消息队列中发送和接收消息
- 如何在远程消息队列中同步和异步发送和接收消息
- 如何调整MSMQ的设置以达到最佳效果
- 如何保证MSMQ CE的可靠性、性能和安全

总论

- MSMQ 为开发人员提供了一组简单方便的API，是CE平台上最简单的一种消息机制
- 对于分布式应用程序之间的松耦合通讯是一种理想的选择
- 在可以连接Internet的Pocket PC / Smartphone上，MSMQ HTTP提供了新的数据交流方式
- 虽然MSMQ本身不提供数据安全机制，但可以通过自定义消息格式的方法来保证数据的安全性

实用工具

- Cambridge Computer Corporation vxUtil
  - Ping, DNS, Route etc…
  - Other tools…
  - http://www.cam.com/windowsce.html
- Marc Zimmerman's Pocket Hosts
  - Create hosts in registry
  - Other tools…
  - http://www.zimac.de/cestuff.htm
- Visual Studio 2005 Device Command Shell
  - Visual Studio "Command Window" add-in
  - Supports remote device commands such as
    - Copying files
    - Starting/stopping/listing processes
    - Working with the device registry
    - Remote XML provisioning and more
  - http://www.gotdotnet.com/workspaces/workspace.aspx?id=50618f79-c7b1-4588-9c0a-cf4ddae8092a

参考资料

- Microsoft Message Queuing Center
  http://www.microsoft.com/windows2000/technologies/communications/msmq/default.asp
- Mark Ihimoyan's Blog: System.Messaging (MSMQ) in .NET CF 2.0
  http://blogs.msdn.com/ihimmar
- C Sharp corner article:
  MSMQ on Pocket PC 2003
  http://www.c-sharpcorner.com/Code/2004/Sept/MSMQonPocketPC2003.asp
- MSDN Magazine:
  MSMQ AND .NET Send MSMQ Messages Securely Across the Internet with HTTP and SOAP
  http://msdn.microsoft.com/msdnmag/issues/03/12/MSMQandNET/default.aspx

获取更多MSDN资源

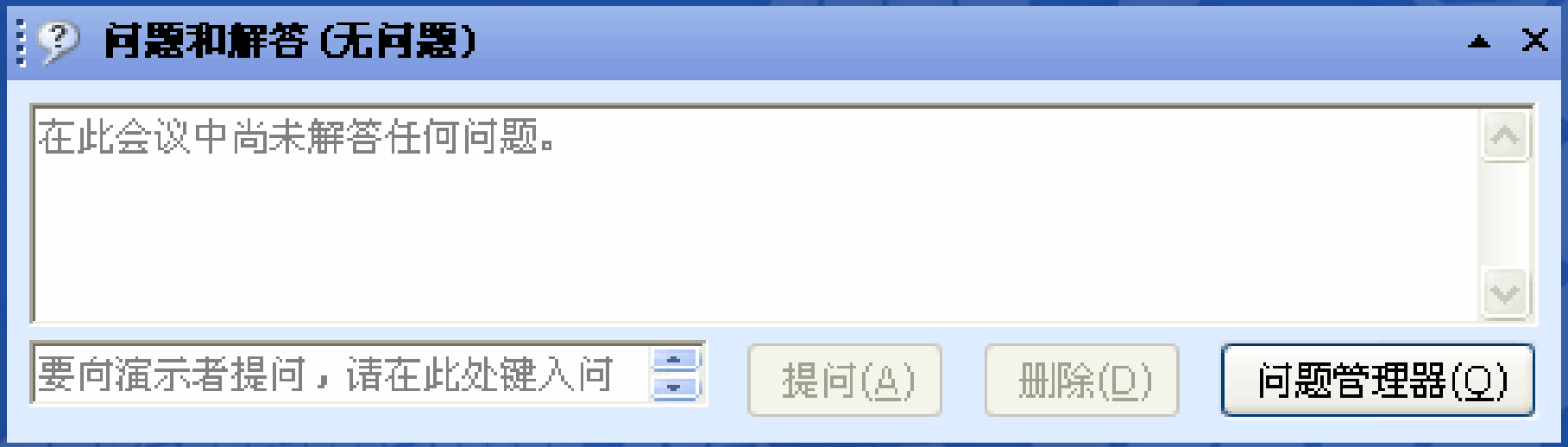
您的潜力，我们的动力

Microsoft
微软（中国）有限公司

- MSDN中文网站
  http://www.microsoft.com/china/msdn
- MSDN中文网络广播
  http://www.msdnwebcast.com.cn
- MSDN Flash
  http://www.microsoft.com/china/newsletter/case/msdn.aspx
- MSDN开发中心
  http://www.microsoft.com/china/msdn/DeveloperCenter/default.mspx

Question & Answer

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。




您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts