Softwired

An Introduction to Wireless JMS

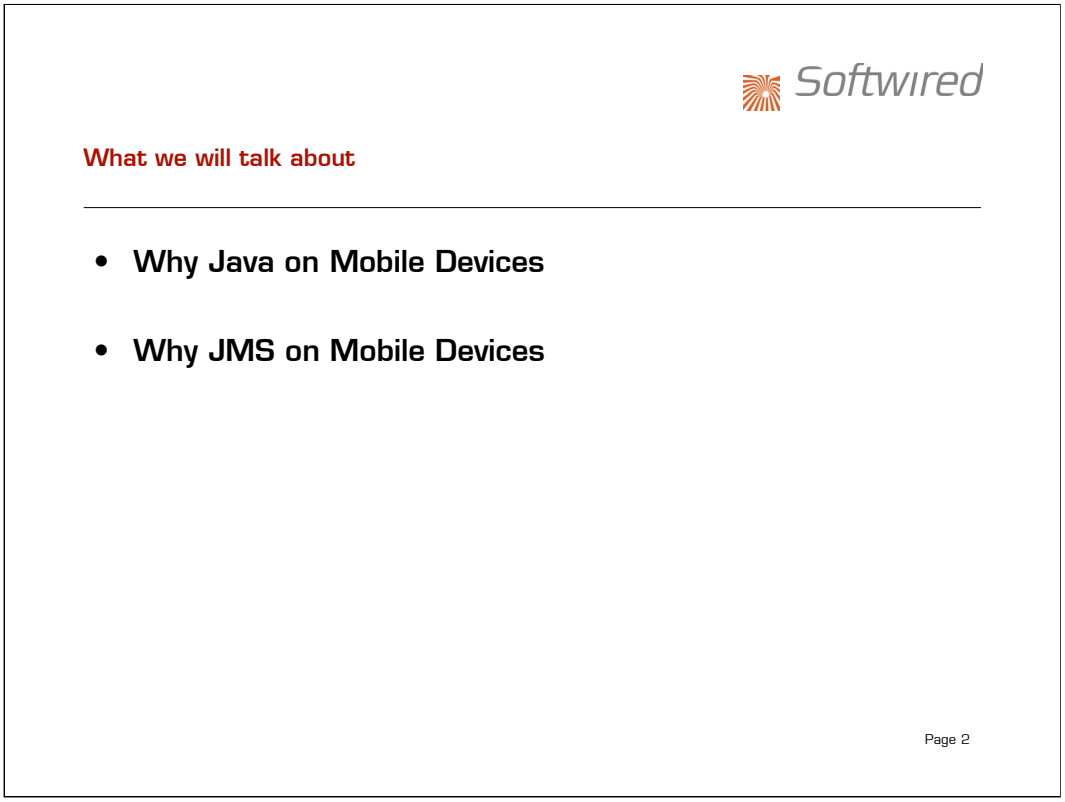
Dr. Silvano Maffeis, CTO

Softwired AG, Zürich

silvano.maffeis@softwired-inc.com

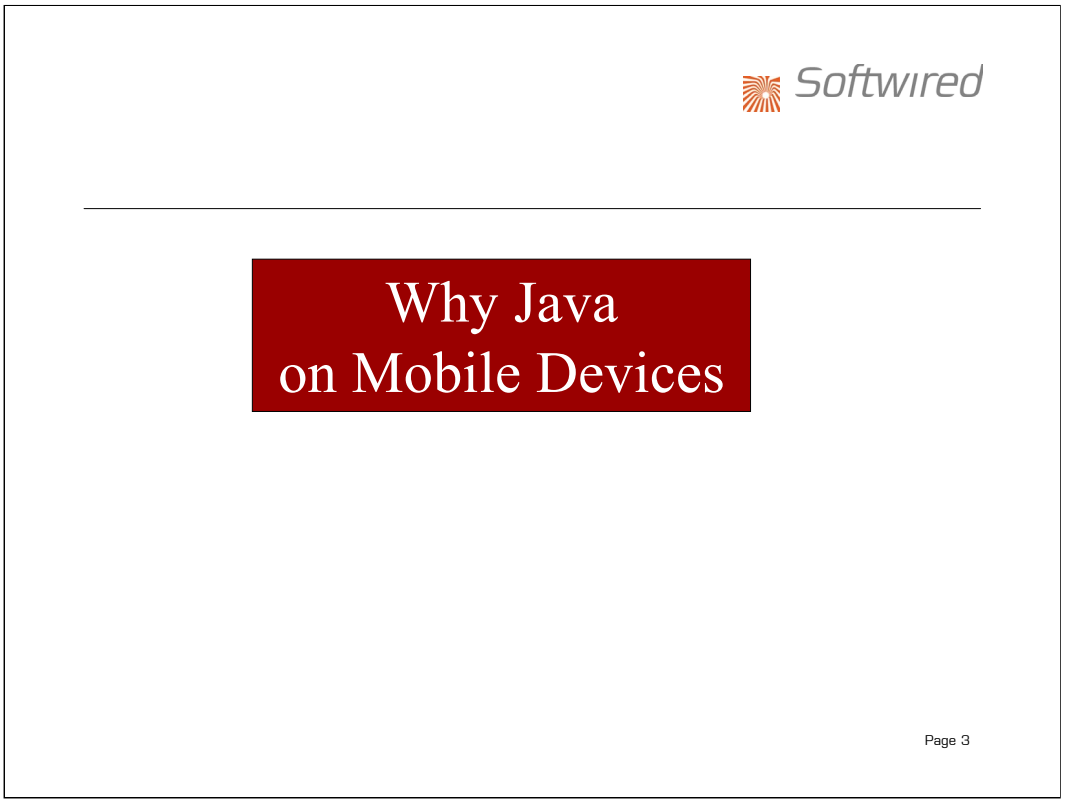
http://www.softwired-inc.com/

Intended audience: Project managers, developers, analysts.



The goals of the presentation are

1) To explain the advantages of deploying customized Java applications on mobile devices, as opposed to a pure micro-browser model (e.g., WAP).
2) To explain the benefits of using Wireless JMS middleware for connecting rich J2ME clients to J2EE "back-ends".



Softwired

Market trends

- **Appearance of communicator devices. (Communicator = Cellular phone + PDA)**
- **Appearance of packet-oriented wireless bearers: GPRS, EDGE, and UMTS.**
- **In 2004, mobile Internet devices will outnumber PCs with Internet access. (TIMElabs, Nomura Research).**

Page 4

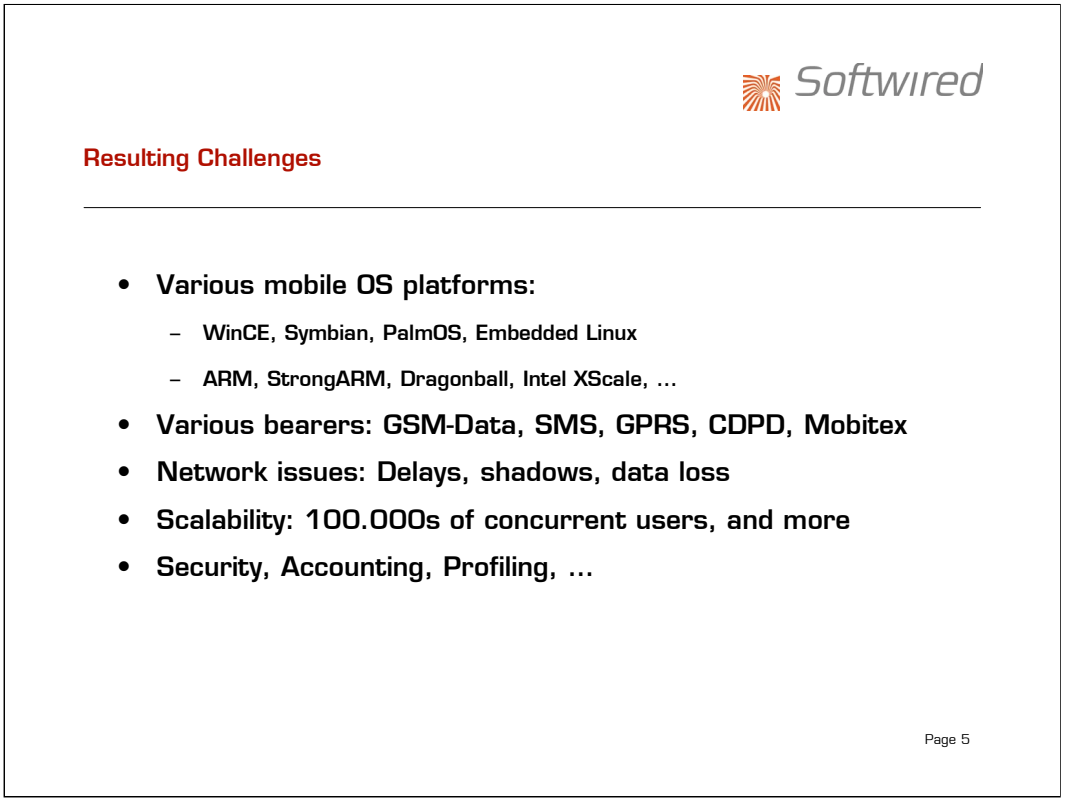
Companies such as Motorola, Nokia, Compaq, HP etc. are investing billions in producing and marketing a new generation of mobile devices; so-called “communicators”.

A communicator is a combination of a PDA and of a mobile phone. It has a high-resolution display, a fast CPU, and built-in 2.5G or 3G wireless data connectivity. Examples: iPaq, HP Jornada, Motorola Accompli, Palm with GPRS Snap-On modem.

At the same time, Telcos have invested billions in 3G licenses (notably UMTS). 2.5G networks (mainly GPRS) are becoming widely available now.

To make this huge investment profitable, new types of mobile services are required. Such which take advantage of the characteristics of communicators, as well as of 2.5/3G bearers, and provide a rich experience to the enduser.

Big corporations are looking at this new technology for making their mobile worker forces more effective.



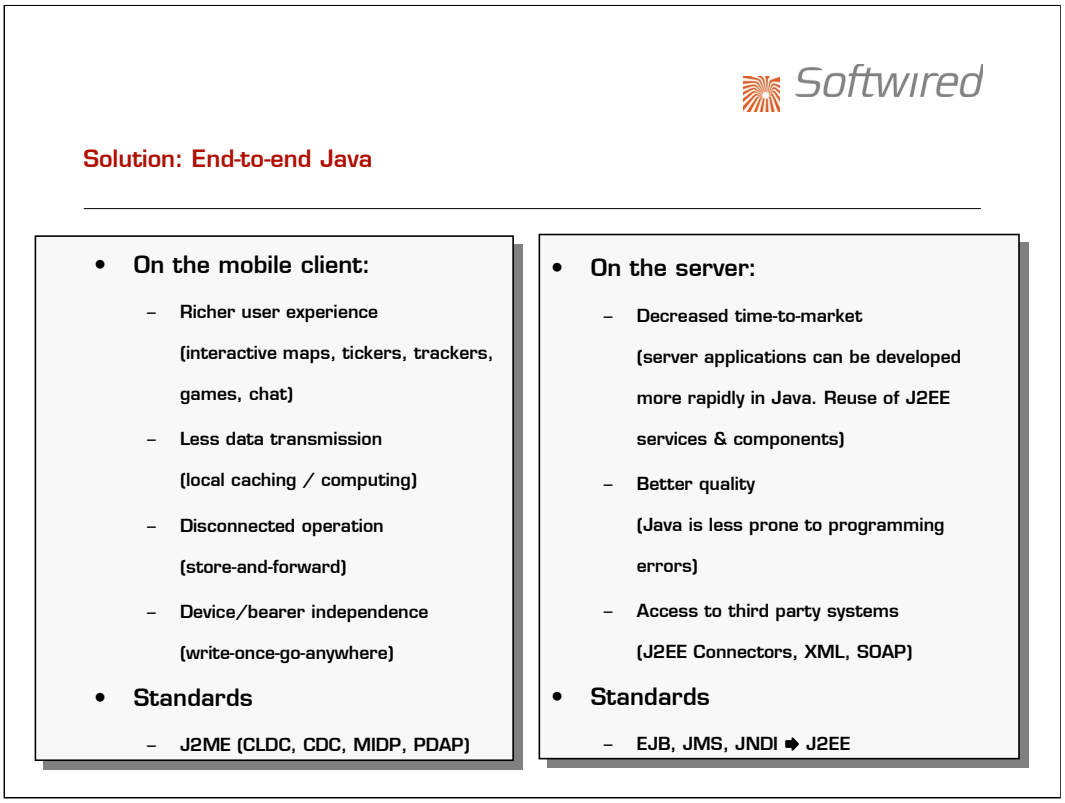
These are the challenges facing developers of mobile services for 2.5/3G.

The mobile platform market is fragmented, it is unlikely that any of the mobile OS platforms available today will dominate the market in the medium or longer term.

To grasp a large user base, mobile services providers therefore need to develop their applications such way that they can run on various platforms, as well as on various wireless networks.

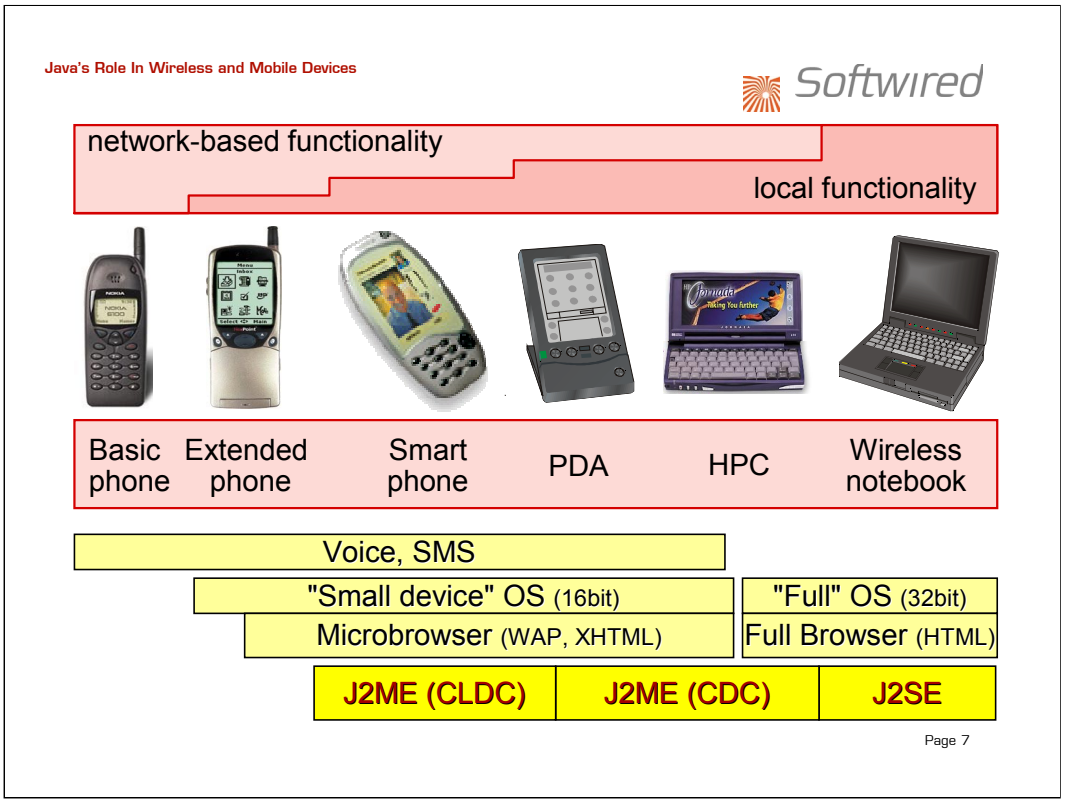
Importantly, wireless networks have some “user-unfriendly” characteristics: Network coverage shadows, drastic variations in available bandwidth, data loss, etc. To convey a positive end-user experience, the illusion of a stable, reliable, and secure network connectivity must be conveyed to the end-user.

Also, transmission delays should be hidden from the end-user as far as possible. This can be achieved through Wireless JMS, as we will explain later.



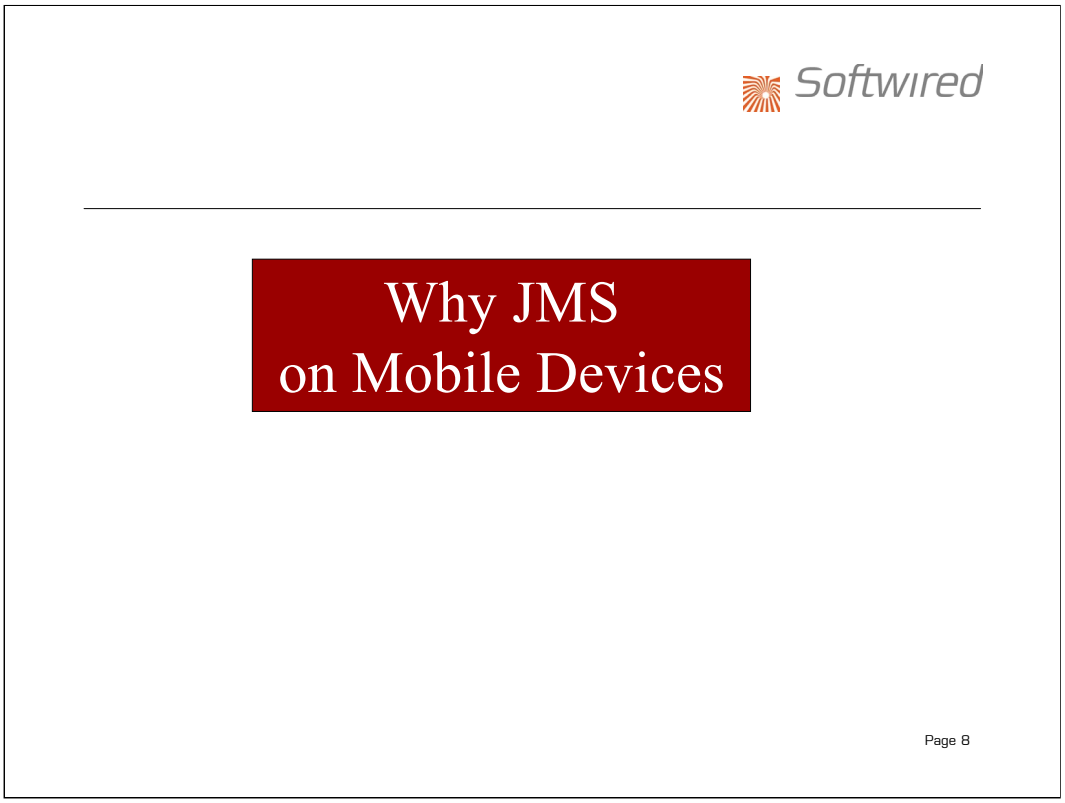
Part of the aforementioned challenges can be addressed by deploying Java on the back-end servers, as well as on the mobile devices (end-to-end Java).

The main motivation of deploying Java applications on mobile devices is "improved user experience": With Java, applications can be provided which are more interactive, which can hide delays and network shadows from the user, and, which are more fun to use than WAP!



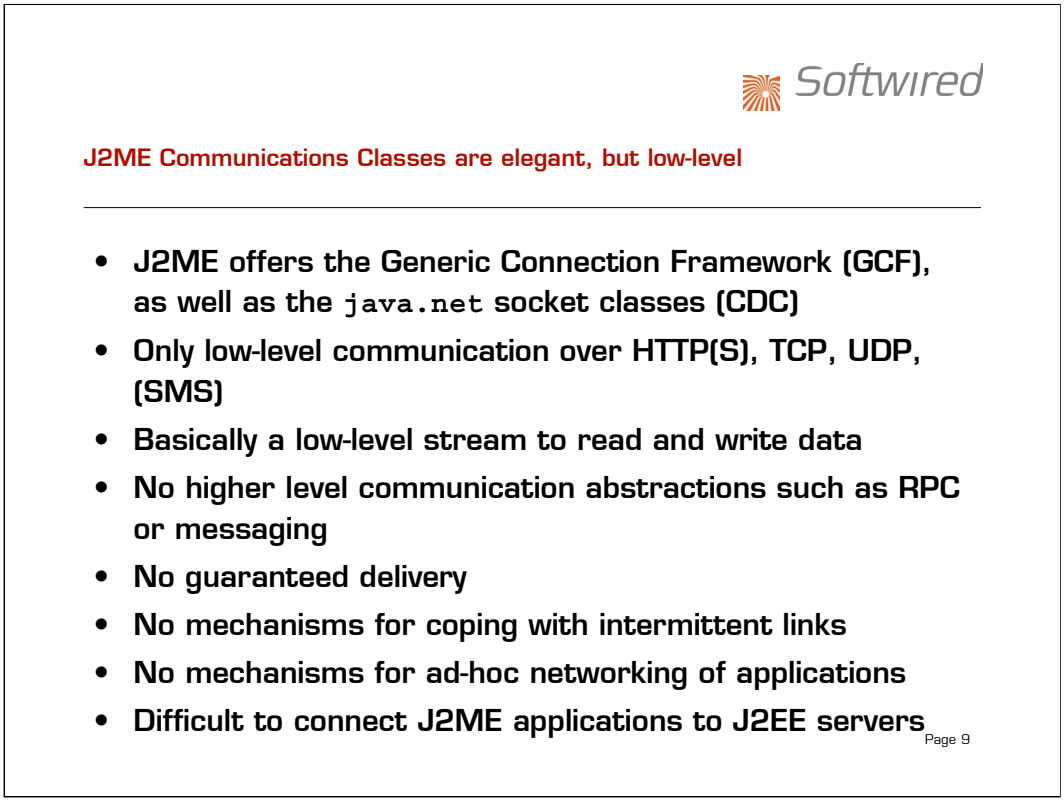
SUN's J2ME platform provides an optimized Java execution environment for each type of device. The depicted devices vary substantially in respect to display size, CPU power, memory size, and communication capabilities.

By putting the right J2ME configuration (CLDC=Connected Limited Device Configuration, or CDC=Connected Device Configuration) plus the right J2ME profile (MIDP=Mobile Information Device Profile, or PDAP=PDA profile) into place, the capabilities of a specific device can be exploited.



We have shown challenges and problems of developing mobile applications. Some of these problems are solved by an end-to-end Java deployment.

But an important component is still missing: wireless middleware for connecting J2ME applications to J2EE back-ends!



The J2ME Generic Connection Framework (GCF) is elegant, but low-level. If we want to convey a good user experience through mobile Java applications, we need guaranteed delivery of transactions ("if the user pays for content, its delivery must be guaranteed to the user"), push notifications, peer-to-peer channels, disconnected operation.

This is NOT provided through the GCF! Building such capabilities atop GCF is a complex and time-consuming exercise.

Softwired

MOM to the rescue

- **MOM=Message Oriented Middleware: message queueing & publish/subscribe**
- **MOM copes well with intermittent communication links: „store-and-forward“**
- **MOM copes well with varying bandwidth and transmission delays: asynchronous messaging, batching, transactions**
- **„Intuitive“ programming model**

Page 10

MOM has been used successfully for over 20 years now. It is the predominant data communications platform in the financial industry, for example. MOM has some unique characteristics, notably store-and-forward transmission of messages, making it very appealing for wireless data communication.

MOM works very well in an environment where network connections sometimes break, and where the available bandwidth can vary dramatically from one moment to another.

CORBA and RMI will NOT work well in such an environment, because this type of middleware was designed to run atop robust network connections (intranet environment).

Softwired

What is JMS?

- JMS stands for "Java Message Service"
- Sun's Definition: *JMS is an API for accessing enterprise MOM systems from Java programs.*
- JMS is for *MOM systems* what JDBC is for *database systems*: A standardized API for accessing them
- JMS consists of an API library (`javax.jms`) and of a run-time infrastructure (message server daemon, tools, ...)
- JMS is part of the J2EE platform
- With J2EE 1.3, every application server vendor must support JMS!

Page 11

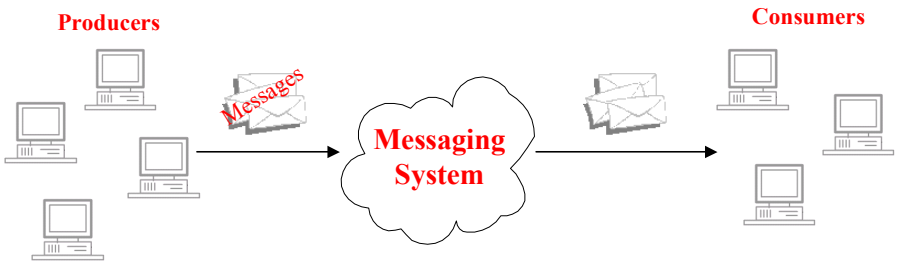
A short introduction to JMS. Importantly, JMS is THE standard for programming MOM applications in Java.

Every J2EE application server must provide JMS.

Softwired

What are Messaging Systems?

- Messaging is a model in which applications are *loosely coupled* through the exchange of *self-describing messages*.
- Crude analogy: E-mail systems for applications



Page 12

Messaging allows producer applications to send messages to consumer applications, in a reliable and scalable manner.

The messaging system decouples producers and consumers. It's in charge of transporting and storing messages.

"Producer" and "Consumer" are just roles an application can play. An application can be both producer and consumer simultaneously, or switch roles dynamically at run-time.

Messaging is to applications what e-mail is to humans. Its often they only way for getting work done efficiently!

Softwired

How does JMS work?

- JMS Domains
  - Publish / Subscribe
  - Point to Point
- JMS Message Flow API
  - Destinations (Topics & Queues)
  - Message Producers
  - Message Consumers
  - Messages (Text-, Map-, Stream, Bytes-, ObjectMessage)
- JMS Control API
  - Connection Factories
  - Connections
  - Sessions

Page 13

JMS provides two communication models: Publish/subscribe, and Point to Point (aka “message queuing”).

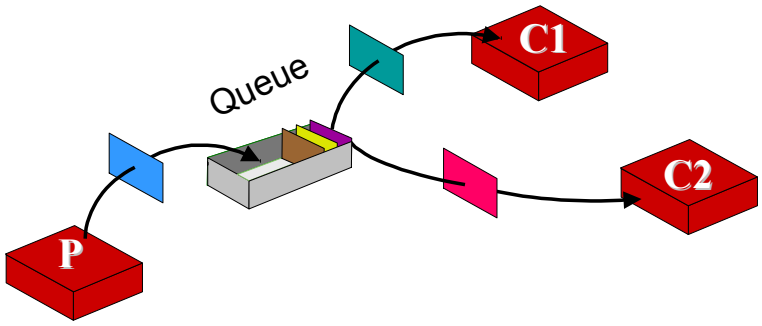
The JMS classes are provided in the Java package javax.jms. These consist of classes for sending and receiving messages (“message flow API”), and of classes for controlling that message flow (“control API”).

(Controlling the message flow means starting and stopping the message flow, issuing transactions, etc.)

Softwired

JMS Point to Point Model

- **One-to-one communication (like e-mail)**



Page 14

In the JMS point to point model, a message is consumed only once. After that the message “disappears” from the queue.

A queue can have multiple consumers, but they will never “see” the same messages (look at the colors).

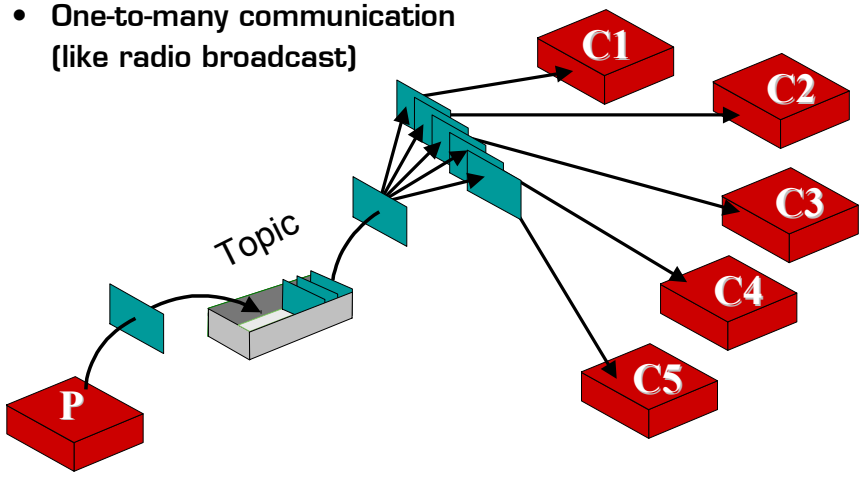
Queues can be used for e-commerce transactions, as an example.

JMS applications can use many different queues simultaneously.

Softwired

JMS Publish / Subscribe Model

- **One-to-many communication (like radio broadcast)**

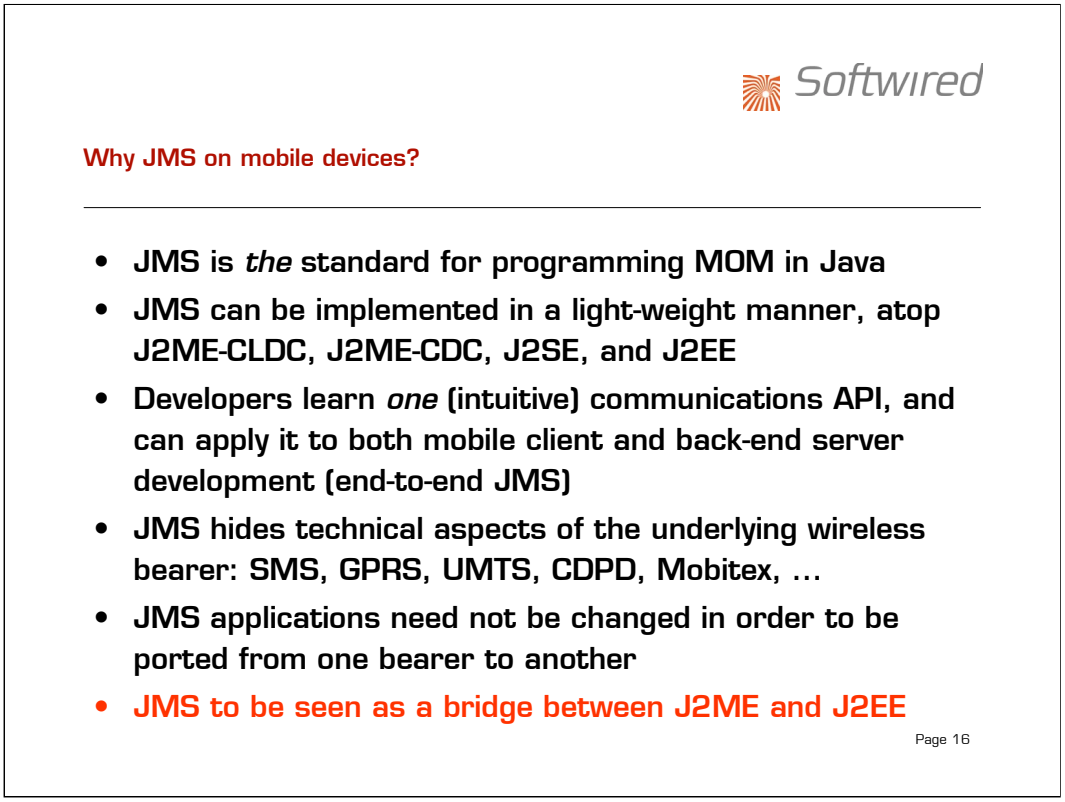


Page 15

Publish/subscribe is a one-to-many multicast communication model. Consumers see the same messages. That's the main difference to the point-to-point model, which is a one-to-one abstraction.

Publish/subscribe can be used for transmitting stock quotes to the workstations on a trading floor, as an example.

JMS applications can use many different topics simultaneously



Now that we understand the basics of JMS, why should one consider using the JMS API for developing mobile Java applications?

Previously we stated that MOM has characteristics which are very appealing for mobile applications (intuitive programming model, latency hiding, guaranteed delivery, store-and-forward). Since we all want to reduce vendor dependence, it's a natural choice to expect full JMS compatibility from a Wireless MOM product!

Softwired

Wireless JMS Components: Softwired's iBus//Mobile Product

- **iBus//Mobile is industry's first Wireless JMS (WJMS) implementation**
- **First commercial version shipped Dec. 2000**
- **iBus//Mobile components:**
  - **WJMS client library for various platforms**
    - J2ME-CLDC (MIDP) (KVM, J9): Palm, JavaPhones, …
    - J2ME-CDC (CVM)
    - PersonalJava, J2SE: iPaq, HP Jornada, Symbian
  - **WJMS Gateway service**
  - **Web admin tool (for system configuration and –monitoring)**

Page 17

Now let's get more "real" by looking at the iBus//Mobile product. iBus//Mobile is the only Wireless JMS implementation available on the market today.

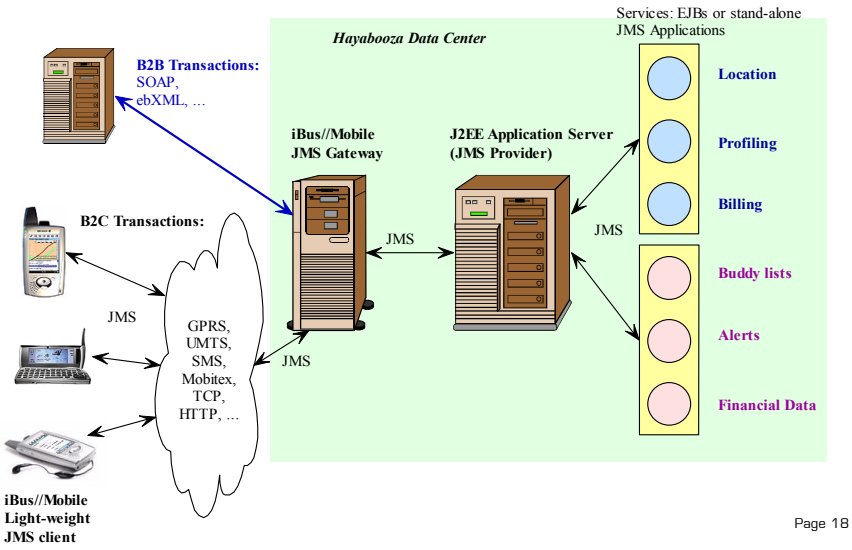
Similar products are typically not JMS compatible (they use proprietary APIs), or not implemented in Java.

iBus//Mobile supports various platforms (mobile OS, Java environments) and wireless networks.

iBus//Mobile can be extended for yet unsupported platforms and networks.

Softwired

iBus//Mobile Deployment



Page 18

This is an architectural overview of a typical iBus//Mobile deployment (mobile portal or WASP).

You can see how iBus//Mobile extends a J2EE application server (WebLogic, WebSphere, iPortal) to become an end-to-end wireless Java platform.

Location-, Billing-, Profiling-, Authentication-services are deployed in the form of EJBs. So also the application specific services (financial data, m-commerce, mobile workforce, ...).

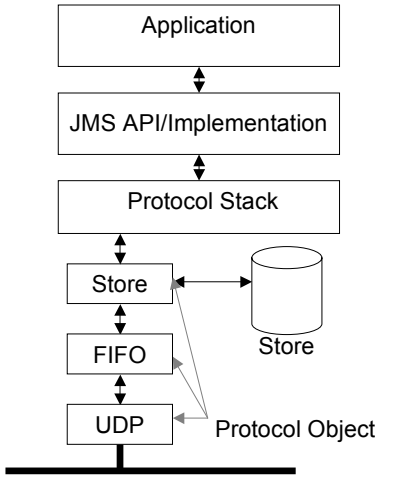
These back-end services use JMS to communicate with each other. The mobile devices use JMS to communicate with other devices, or with the back-end services.

This results in an elegant and manageable platform, on which rich applications can be developed more quickly and easily!

Softwired

iBus//Mobile WJMS Client Library

- **Mobile application uses *standard* JMS API**
- **Ligh-weight JMS library deployed on mobile devices**
- **A local database (on the device) is used for store-and-forward**
- **Supports off-line operation**
- **Supports various bearers**
- **Security: SSL, Certicom, PKI, …**
- **Builds upon the J2ME Connection Framework (Stream, Datagram)**



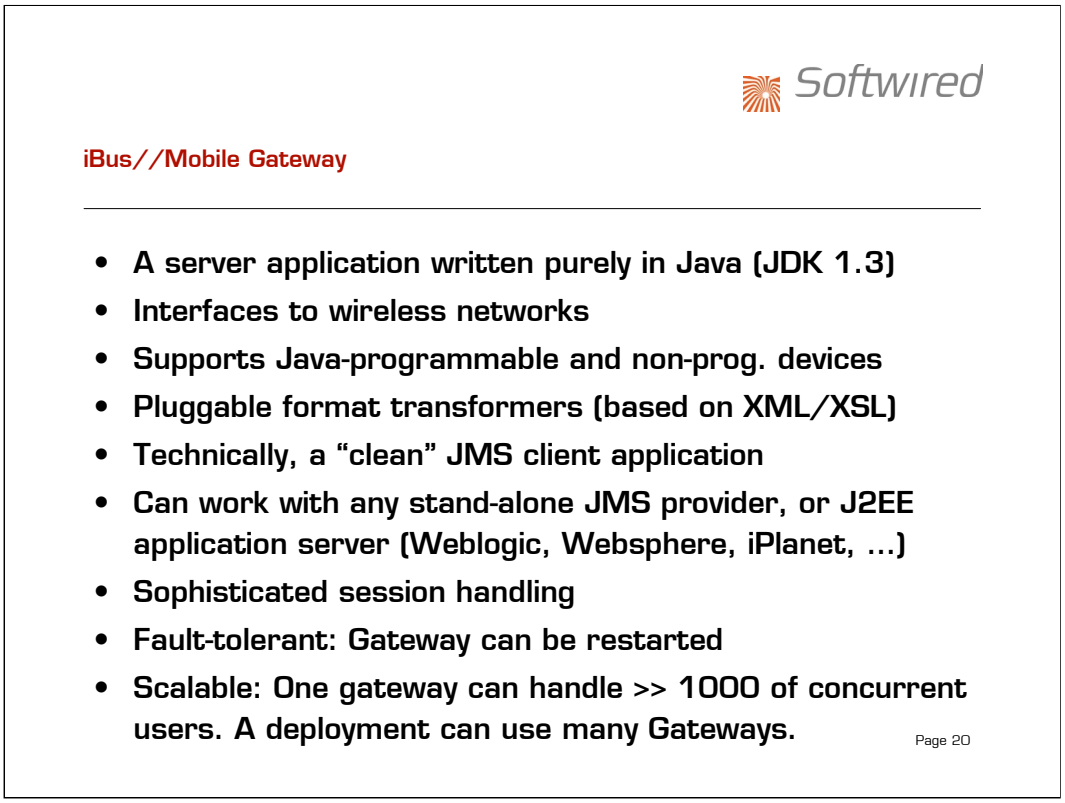
Page 19

This shows the design of the iBus//Mobile Wireless JMS client library. Owing to the iBus//Mobile protocol stack design, iBus//Mobile can run atop virtually any wireless network (GSM Data, SMS, GPRS, UMTS, CDPD, Mobitex, …) or communications protocol (TCP, HTTP, UDP).

Any protocol exposed through the J2ME Generic Connections Framework can be used by iBus//Mobile, no matter whether it's a Stream- or a DatagramConnector.

iBus//Mobile can use TCP/IP for reliability. But TCP/IP is often not efficient enough on wireless networks (slow starts, expensive connection set up, no "unsolicited" push from server to client).

Therefore, iBus//Mobile comes with its own reliability protocol optimized for wireless. This protocol can be used directly on top of GPRS, without necessarily requiring TCP or HTTP.

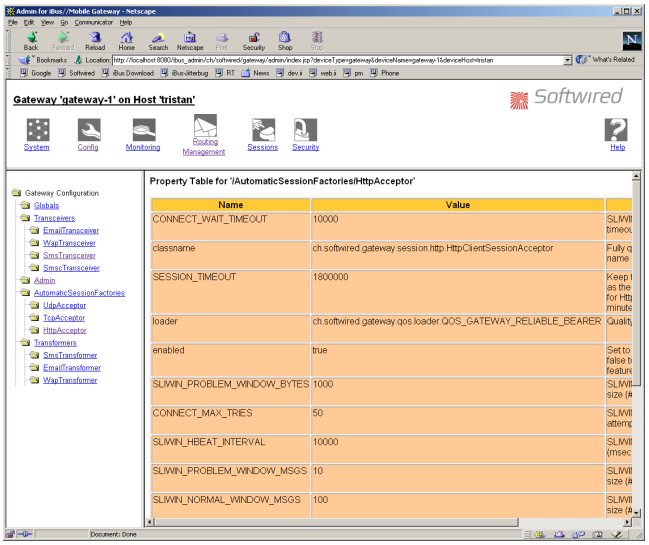


The gateway is an invisible but very important component of iBus//Mobile. It's a quite sophisticated piece of Java server software that does session management for the mobile clients, access control, message transformation, message routing, etc.

The gateway is a "clean" JMS application in the sense that it imports only javax.jms, and in that it does not use any Softwired proprietary APIs. This is very important as it allows iBus//Mobile to support other JMS providers, besides iBus//MessageServer. E.g., the JMS provider that is part of a J2EE application server. Or any stand-alone JMS provider.

Softwired

iBus//Mobile Administration Tool



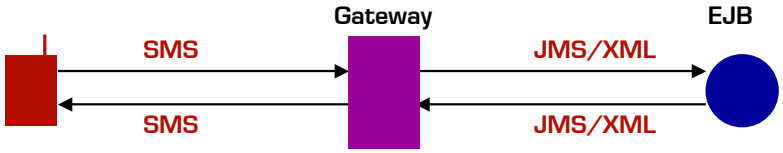
This is a screenshot of the iBus//Mobile administration tool. A web browser is used to connect to the iBus//Mobile gateway.

Through the administration tool, you can shutdown and restart the gateway, or see which Wireless JMS applications are connected to the gateway. You are also able to see on what platform (JDK, mobile operating system, wireless network) a WJMS application is running.

Other features include the configuration of the various wireless adapters that can be plugged into the gateway, activity logging, security configuration, etc.

Softwired

Interaction Scenario: SMS phone or pager



- Conventional SMS (or "page") is sent to iBus//Mobile Gateway
- Gateway converts the SMS into JMS, and puts it into a Topic or Queue (configurable)
- Message bodies encoded in XML
- "Back-end" (EJB or standalone JMS app) processes message
- Back-end sends a response JMS message
- Pluggable format converters: SMS, WAP, E-Mail, FTP, ...
- Back-end developer only needs to know JMS!
- Decoupling, reliability, store-and-forward

Page 22

Interaction scenario. SMS to Gateway, gateway converts SMS into XML, and puts that XML into a JMS message. The JMS message is put into a JMS queue or topic.

The backend reads the message from the queue (or topic), processes it, and sends back a response to the mobile device.

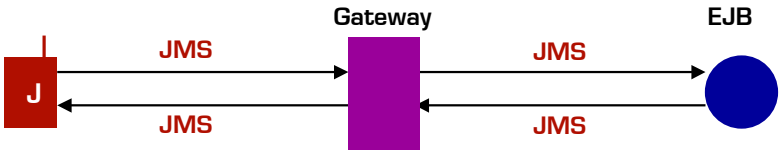
The backend can be written in a standard manner using the JMS or EJB APIs, without particular knowledge of SMS. This is important, because most of the SMS services available today are written using proprietary C or C++ libraries. These solutions are tied to a particular vendor, and to SMS! (No portability).

With iBus//Mobile, the same backend can be used along with other messaging bearers (MMS, Mobitex, paging, e-mail, ...).

Also, reliability is increased. If the backend (EJB) is not running when an SMS arrives, the JMS message is stored inside the JMS provider until the backend becomes available. (Decoupling).

Softwired

Interaction scenario: JavaPhone or PDA



- Client device is Java programmable (CLDC-MIDP, CDC, PersonalJava, J2SE)
- Client device uses iBus//Mobile WJMS library
- Full JMS API available to mobile applications
- WJMS library encodes JMS message efficiently for wireless
- WJMS library implements optimized reliability protocols
- Gateway converts WJMS messages into JMS messages suitable for JMS provider (iBus//MessageServer, Weblogic JMS, JMQ, ...)

Page 23

That's the end-to-end JMS scenario. Use one and the same API on the backend and on the mobile devices. Developers only need to know the names of Topics and Queues they want to use.

iBus//Mobile implements the illusion of JMS Destinations which are available everywhere.

All the rest (reliable message transmission, store-and-forward, brokering, etc.) is done by iBus.

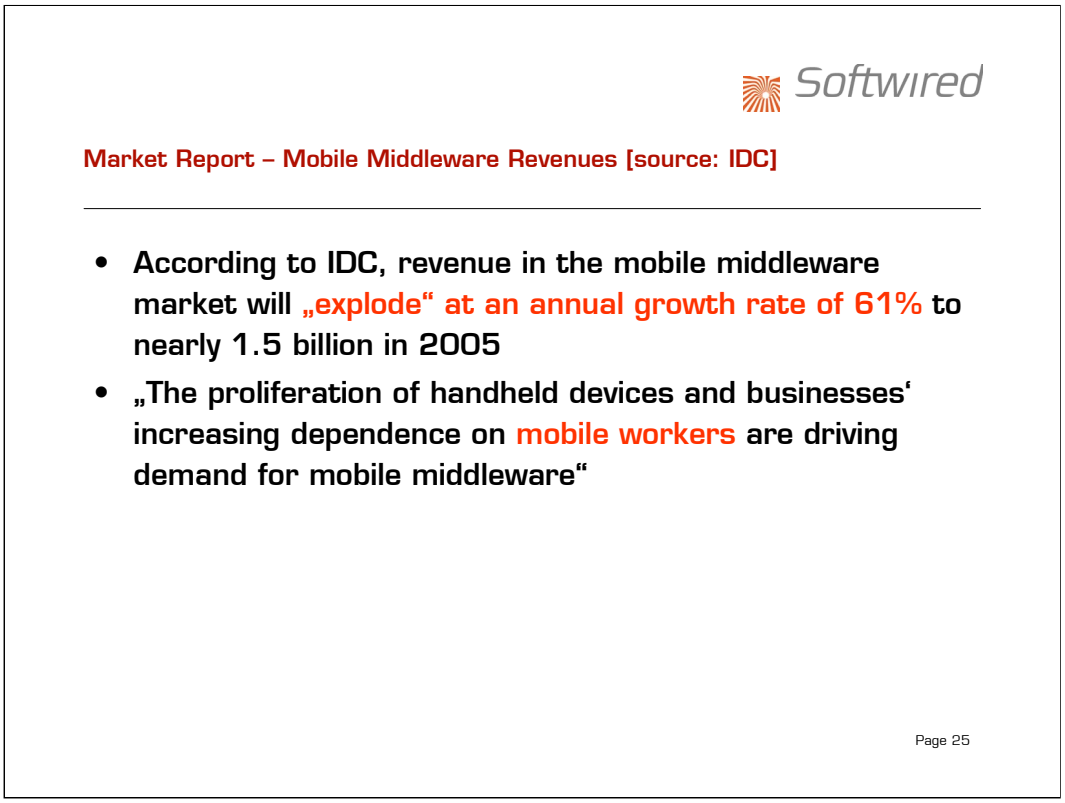
Softwired

WJMS Benefits

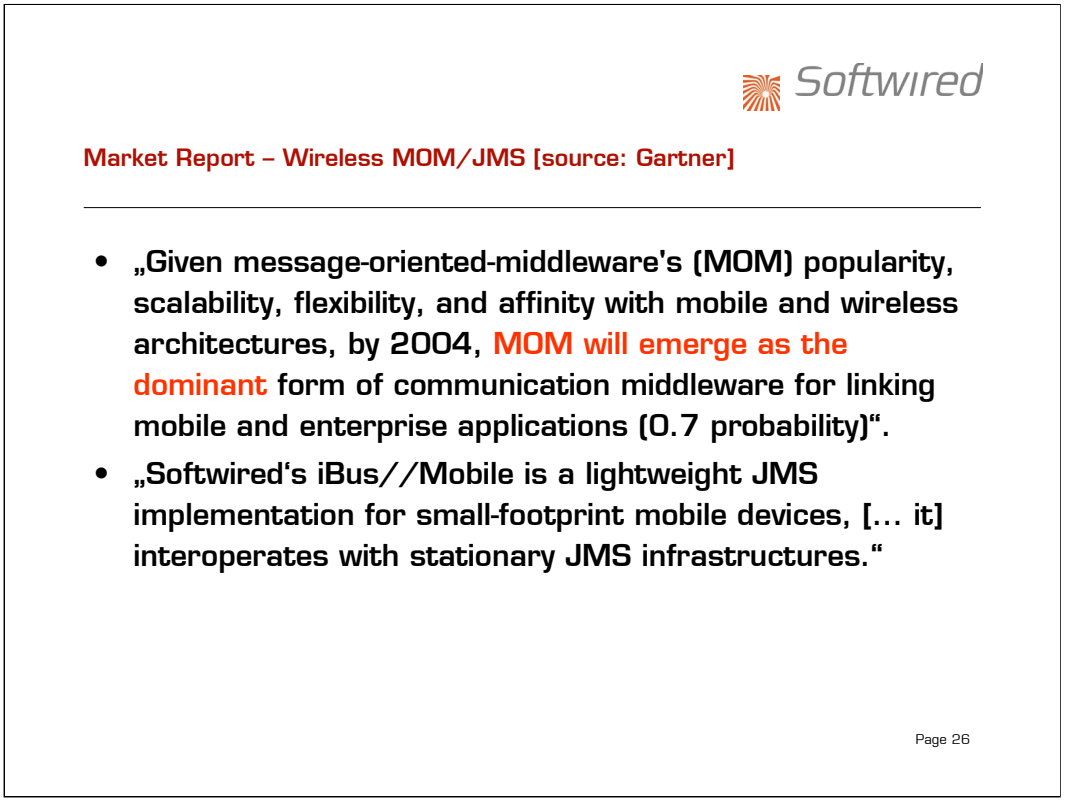
- JMS Topics and Queues accessible on devices as well as on back-end: End-to-end solution, "ubiquity"
- One programming model
- Applications are written against JMS API, instead of low-level wireless API
- Write once – Run on different wireless networks!
- Spontaneous networking
- Applications are always "in sync". No extra data synchronization middleware required
- Rich interaction models (push, peer-to-peer, transactions, ...) means better user experience
- WJMS applications can be operated in offline mode
- WJMS "hides" network shadows, delays, etc.

Page 24

A recap of the WJMS benefits.



This is a recent analyst report about the wireless middleware market. IDC=International Data Corporation. A well known analyst company in the IT area. The market outlook for wireless middleware is very promising!



This is another recent analyst report about the wireless middleware market. Gartner is probably the best known analyst company. They narrow the wireless middleware market down to wireless MOM. This implies that CORBA and RMI are of less appeal for mobile computing.

Gartner says Softwired is well positioned to be a successful player in this market.

Softwired

Conclusions

- For mobile application development, Java is a compelling choice: Platform neutrality, ease of development, client provisioning, user experience!
- For mobile application data exchange, JMS is a compelling choice: Disconnected operation, reliability, scalability
- The combination of J2ME, J2EE application servers, and JMS leads to a standards-based mobile Java applications platform
- Selling arguments: Richer user experience, reduced time to market, reduced risk!

Page 27

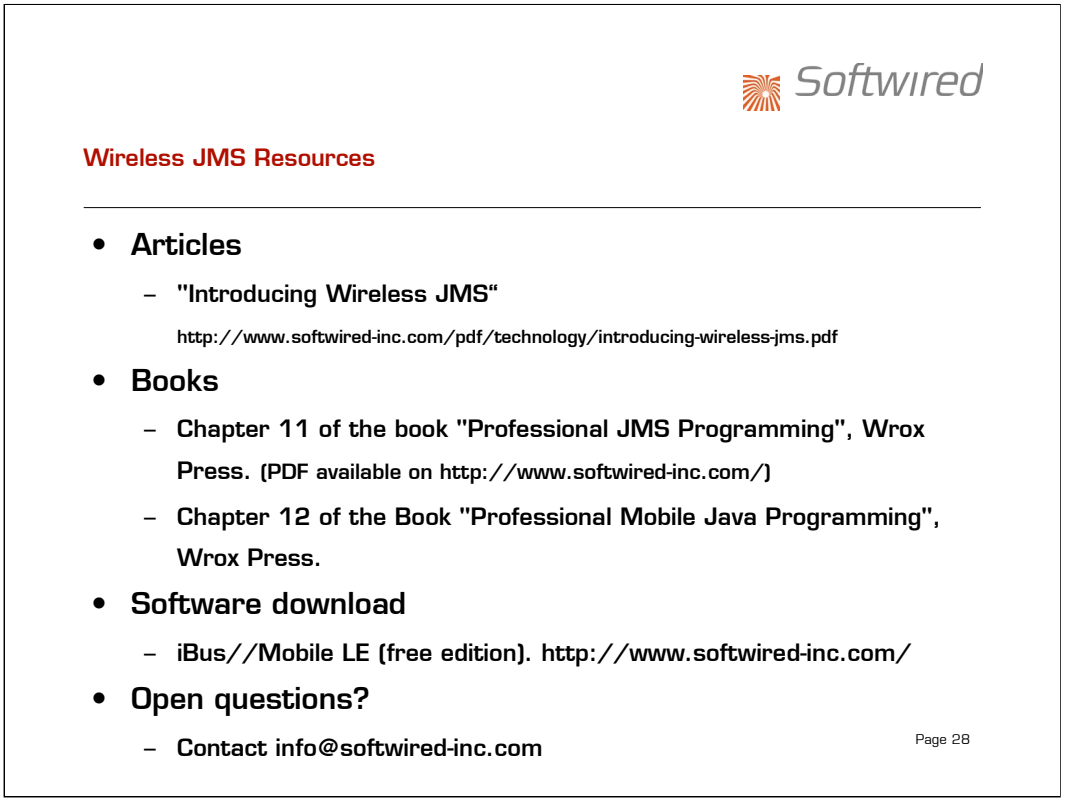
We have given many **technical** reasons why WJMS is an important and necessary component. However, the **non-technical** benefits of WJMS are even more important, as these impact the end-user:

-Better user experience, and more fun, due to high interactivity

-Again, better user experience as WJMS hides the "user-unfriendly" aspects of wireless networks: delays, shadows, broken network connections, uncertainty whether transactions succeeded or not.

-Better time to market due to faster development and portability of WJMS applications across platforms (OS, wireless network)

-Less risk by following open standards (J2ME, J2EE, JMS).



Here you will find more information about Wireless JMS.