Archive
Auto Service Request (ASR)
All System Admin Articles
All Systems Topics
Cool Threads
DST
End of Notices
FAQ
Hands-On Labs
High Performance Computing
Interoperability
Patches
Security
Software Stacks
Solaris Developer
Solaris HowTo
Solaris Studio IDE Topics
Sysadmin Days
System Admin Docs
Upgrade
VM Server for SPARC
Did you Know
Jet Toolkit
Oracle ACES for Systems

Getting Started With the PIM APIs

By Qusay Mahmoud, February 2005

Managing Personal Information on Mobile Devices Using JSR 75

Personal information management (PIM) refers to the ability to manage in electronic form the kinds of personal data that broad classes of users want handy, such as appointment books, contact directories, and to-do lists. Having this information literally at your fingertips, wherever you go, is a strong motive to buy a mobile device. PIM data is often stored on the device in a native format, intended for access by applications the vendor provides. Some vendor-independent standards have been developed. The Internet Mail Consortium manages two that facilitate the electronic exchange of contact and calendar data with other PIM applications, and its transmission over the Internet: *vCard,* a business card format, and *vCalendar,* a scheduling-exchange format.

Until recently, developers of applications based on the Java 2 Platform, Micro Edition (J2ME) have found access to device-based PIM difficult, because the Connected Limited Device Configuration (CLDC) and the Mobile Information Device Profile (MIDP) do not themselves define APIs for easy access to native databases, or to data in the vCard and vCalendar formats. Now, however, JSR 75, the PDA Optional Packages for the J2ME Platform defines an API that gives J2ME devices that implement the specification access to native PIM data on devices, and to contact and scheduling data in the vCard and vCalendar formats.

This article provides a code-intensive introductory tutorial to the PIM APIs; it:

Introduces JSR 75

Describes the `javax.microedition.pim` optional package

Provides details about the PIM APIs

Offers a taste of the effort involved in using them

Provides code that you can adapt to the needs of your own wireless applications

The PDA Optional Packages for the J2ME Platform

JSR 75 defines two optional packages that extend and enhance CLDC to give J2ME developers access to important features commonly found on PDAs:

The FileConnection (FC) APIs give J2ME devices standardized access to file systems residing on mobile devices, primarily to allow access to removable storage media such as external memory cards. For a tutorial on this package, see the article " Getting Started with the FileConnection APIs."

The Personal Information Management (PIM) APIs optional package gives J2ME devices access to personal data stored in device-native formats, and is the subject of this article.

Any device that supports CLDC 1.0 can support JSR 75, which means that the PIM APIs can be deployed on top of any J2ME profile that supports CLDC 1.0 or higher. Because the Connected Device Configuration (CDC) is a superset of CLDC, the PIM and FC optional packages can be deployed on both CLDC- and CDC-based devices. Note that both packages are optional, and that the two are independent of each other.

Who has the option? The decision whether to support an optional package on a given device is made by whoever is responsible for maintaining the platform software, usually the device manufacturer. In general, neither an end user nor an application developer can download an optional package and install it in the device. An application that relies on an optional package will not run unless the manufacturer first integrates the optional package into the device.

The PIM Optional Package

The primary objective of the PIM APIs is to give J2ME devices access to the same personal data to which native applications already have access. Core goals of this optional package are to:

Provide access to personal data in native formats, which may reside on the device, on removable media, or somewhere over the network

Support the import and export of address-book entries in vCard format, and calendar and to-do entries in vCalendar format.

Impose no required fields or attributes

Provide security in using these APIs

The PIM optional package API defines three types of PIM data, known as PIM lists:

Contact lists, which contain names, addresses, phone numbers, and other info about business and personal contacts

Event lists, which store appointments, reminders, and other date-specific items

To-do lists, which record tasks that the user needs to accomplish

Not all devices support all three types, but a device must support at least one if the vendor claims that it supports the PIM optional package.

The PIM APIs

The PIM APIs are defined in the package `javax.microedition.io.pim`. This package consists of eight interfaces and six classes, including four exception types:

Interfaces	Description
`PIMItem`	The common superinterface of an item to be stored in a PIM list, where an item is a collection of data for a single PIM entry: a `Contact`, `Event`, or `ToDo`
`PIMList`	The common superinterface of `ContactList`, `EventList`, and `ToDoList`, each of which may contain zero or more `PIMItems`
`Contact`	A single entry in a contact database; the static fields in this interface are a subset of the fields defined by the vCard specification
`ContactList`	A list of `Contact` items
`Event`	A single entry in an event database
`EventList`	A list of `Event` items
`ToDo`	A single item in a to-do database
`ToDoList`	A list of `ToDo` items

Classes	Description
`PIM`	Provides a collection of static methods to find information about and gain access to `PIMLists`
`RepeatRule`	The description of a pattern of repetition for an `Event` item, to specify when the associated event will occur; the static fields in this class are a subset of the capabilities of the `RRULE` field in `VEVENT`, defined by the vCalendar 1.0 specification

FieldEmptyException	Thrown when an attempt is made to access a field that doesn't have any data values associated with it
FieldFullException	Thrown when an attempt is made to add data to a field whose available data-value slots have already been assigned
PIMException	Thrown by the PIM classes
UnsupportedException	Thrown when the field referenced is not supported in the PIM list that an element belongs to

The PIM optional package may not be available on all J2ME platforms. To find out whether any given device implements it, invoke `System.getproperty()` with a key of `microedition.pim.version`. The method will return the version number of the API if it's supported, or `null` if it's not.

> **Note**: CLDC permits implementations to refuse to load an application that refers to classes that are not present, so it may be impossible to perform the runtime check for presence of the PIM APIs on a device that doesn't support them. In this case, you must be prepared to package two different versions of the application: one that uses the PIM APIs and one that doesn't.

Security Concerns

It is the responsibility of the J2ME platform where the PIM APIs are deployed to provide a security model that supports the security of the PIM APIs. Implementation is left to the including profile or platform, but there are two specific requirements:

The security model must support rights to perform read-only, write-only, and read-write access.
The security model must be applied in methods in the `javax.microedition.pim` package that throw `SecurityException`.

Using the PIM APIs

The entry point into data access is through the `PIM` class, which provides a factory method for obtaining a PIM instance:

```
import javax.microedition.pim.*;

PIM singleton = PIM.getInstance();
```

In addition, `PIM` provides functions for import and export; for example, `fromSerialFormat()` and `toSerialFormat()`.

All PIM lists are represented by the `PIMList` interface and its three subinterfaces: `ContactList`, `EventList`, and `ToDoList`.

To get access to a PIM list, you use the method `openPIMList()`. The first parameter identifies the list type: `PIM.CONTACT_LIST`, `PIM.EVENT_LIST`, or `PIM.TODO_LIST`. The second parameter specifies the mode: `PIM.READ_ONLY`, `PIM.READ_WRITE`, or `PIM.WRITE_ONLY`. The third, optional parameter specifies the name of the PIM list desired. If you omit this parameter `openPIMList()` will return the default PIM list, the list of the specified type that the built-in PIM application uses. If you specify a list that the method can't find, it throws a `PIMException`.

```
...
PIM pim = PIM.getInstance();
ToDoList list = null;

try {
   list = (ToDoList) pim.openPIMList(PIM.TODO_LIST, PIM.READ_ONLY, "list-name");
   // use the list
} catch (PIMException pe) {
  // no such list
} catch (SecurityException se) {
  // MIDlet is not allowed access to the specified list
}
...
```

A PIM list contains items represented by the `PIMItem` interface and its subinterfaces `Contact`, `Event`, and `ToDo`. You can use the list's `items()` method to obtain an `Enumeration` of the items in the PIM list:

```
...
Enumeration enum = list.items();
while(enum.hasMoreElements()) {
    ToDo task = (ToDo) enum.nextElement();
    // do something with task
}
...
```

The next example demonstrates how to open a contact list, and how to retrieve and add contact values:

```
...
// Open default contact list
PIM pim = PIM.getInstance();
ContactList clist;
try {
   clist = (ContactList) pim.openPIMList(PIM.CONTACT_LIST, PIM.READ_WRITE);
} catch(Exception e) {
   // security or other exception
}

// Retrieve contact values
// The countValues() method returns the number of data values currently
// set in a particular field.
Enumeration contacts = clist.items();
Contact c = (Contact) contacts.nextElement();
int phoneNumbers = c.countValues(Contact.TEL);
for(int i = 0; i < phoneNumbers; i++) {
   if((c.getAttributes(Contact.TEL) != 0) & Contact.ATTR_HOME != 0) {
      // Home number
      String home = c.getString(Contact.TEL, i);
   }
}

//Add contact values
Contact c = clist.createContact();
int attrs = Contact.ATTR_HOME;
c.addString(Contact.TEL, attrs, "416-799-1313");
// Some fields can be added without attributes
c.addString(Contact.ORG, PIMItem.ATTR_NONE, "someName Corporation");
// Add the item to the native contact database
c.commit();
...
```

Implementations of JSR 75

The official reference implementation (RI) of the PIM APIs is available from IBM at PDA Optional Packages. Because the target of this RI is the PocketPC operating system, it requires the J9 Java Virtual Machine for the PocketPC.

The J2ME Wireless Toolkit 2.2 from Sun Microsystems includes an implementation of JSR 75. I used this toolkit to test the examples in the rest of this article.

Important notes about PIM implementations:

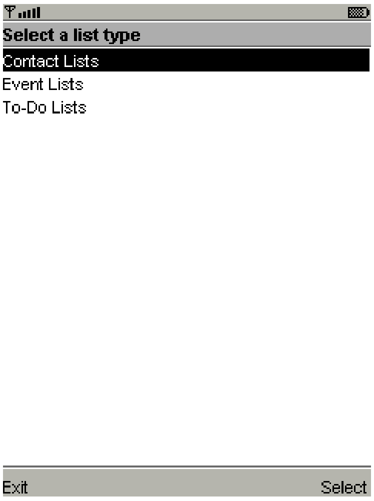
Implementations may provide access to PIM databases that reside at well-known locations such as SIM cards attached to the device and remote PIM databases. As a result, there can be multiple PIM lists of the same type; for example, a cell phone might have two `ContactList`s: one in the phone, and one on the SIM card. These lists will be differentiated by name, and you can retrieve the names by calling `PIM.listPIMLists()`, which returns a list of all PIM list names of a given type.
While the PIM APIs define three types of PIM lists, an implementation can be deemed compliant if it supports only one or two. An implementation must support vCard 2.1 if it supports contact lists, and vCalendar 1.0 if it supports either event or to-do lists. If any type of PIM list is not supported, the methods used for access to lists of the unsupported type must throw a `PIMException`.
Finally, the PIM APIs provide no locking operations. Therefore, implementations should ensure that all PIM list-record operations are atomic, synchronous, and serialized, so that multiple accesses don't corrupt the data. If your application uses multiple threads for access to a PIM list, it is your responsibility to coordinate the access to avoid unintended consequences.
It's also worth noting that not all devices support all fields. You might for example import a vCard that has fields that cannot be stored in a native database. You can test for this problem; the `PIM.isSupportedField()` method indicates whether a given PIM list field is supported. There may not be a general solution to such problems, but at least you can be aware of them before any information is lost.

PIM Demo in J2ME Wireless Toolkit 2.2

The J2ME Wireless Toolkit 2.2 comes with a PIM demo that allows you to manage contact, event, and to-do lists. To experiment with this demo:

Download and install the toolkit if you haven't already.
Start KToolbar.
Open the project, `PDAPDemo`.
Run the application, and you'll see something like Figure 1.

Figure 1: The J2ME Wireless Toolkit's PDAPDemo



Select any of the list types – but don't expect much output. The emulator will not display any lists because it doesn't yet include any.

> **Note**: In the toolkit's emulator, the PIM APIs give MIDlets access to PIM lists stored on your desktop computer's hard disk, in `*toolkit*/appdb/DefaultColorPhone/pim/contacts/Contacts` , `*toolkit*/appdb/DefaultColorPhone/pim/events/Events` , and `*toolkit*/appdb/DefaultColorPhone/pim/todo/To Do` . Contacts are stored in vCard format, and calendar and to-do items are stored in vCalendar format.

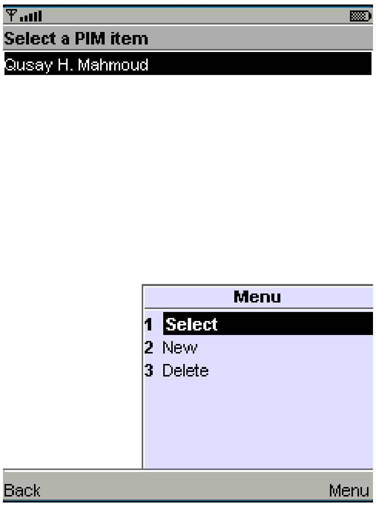
Setting Up PIM Lists

If you have your own vCard, save it in a file in `*toolkit*/appdb/DefaultColorPhone/pim/contacts/Contacts` . If you don't, save the following sample vCard in `sample.vcf`:

```
BEGIN:VCARD
VERSION:2.1
N:Mahmoud;Qusay;H.
FN:Qusay H. Mahmoud
ORG:JavaCourses
TITLE:Developer
NOTE:Helping You With Java!
TEL;WORK;VOICE:(416) 999-1111 ext 1000
TEL;WORK;FAX:(416) 333-9999
ADR;WORK;ENCODING=QUOTED-PRINTABLE:;;1500 Java Ave.=0D=0ASuite 101;Toronto;ON;M9V1L1;Canada
LABEL;WORK;ENCODING=QUOTED-PRINTABLE:1500 Java Ave.=0D=0ASuite 101=0D=0AToronto, ON M9V1L1=0D=0ACanada
URL;WORK:http://www.javacourses.com
EMAIL;PREF;INTERNET:qmahmoud@javacourses.com
REV:20043335T124220Z
END:VCARD
```

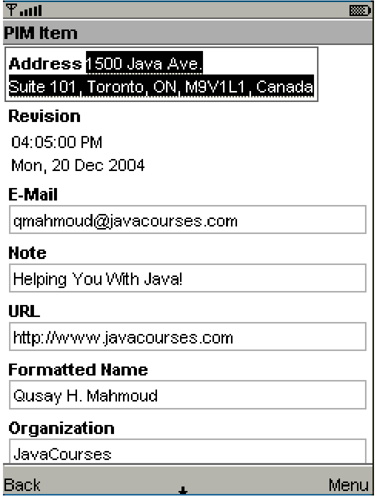
Now if you select **Contact Lists** from the screen shown in Figure 1 you'll see something like Figure 2.

Figure 2: A Contact Item, Stored as a vCard



As you can see, the demo lets you view the selected contact item, delete it, or create a new one. If you select the highlighted item you'll see the full details, as read from the vCard, as in Figure 3.

Figure 3: Access to vCard Information



Address 1500 Java Ave. Suite 101, Toronto, ON, M9V1L1, Canada

Revision
04:05:00 PM
Mon, 20 Dec 2004

E-Mail
qmahmoud@javacourses.com

Note
Helping You With Java!

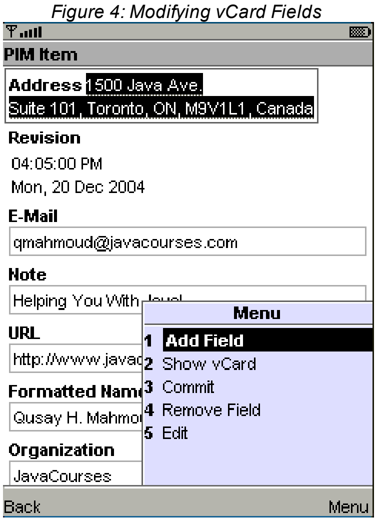
URL
http://www.javacourses.com

Formatted Name
Qusay H. Mahmoud

Organization
JavaCourses

The demo also allows you to edit the fields, as Figure 4 shows.

Figure 4: Modifying vCard Fields



Address 1500 Java Ave. Suite 101, Toronto, ON, M9V1L1, Canada

Revision
04:05:00 PM
Mon, 20 Dec 2004

E-Mail
qmahmoud@javacourses.com

Note
Helping You With Java!

URL
http://www.javac

Formatted Name
Qusay H. Mahmo

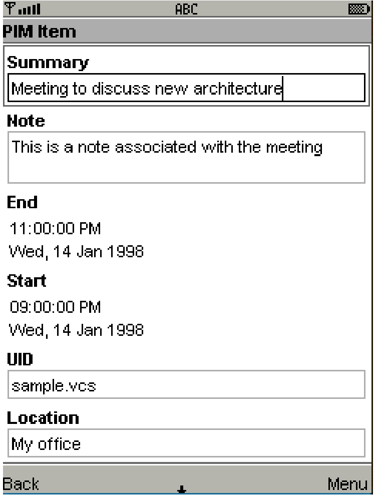
Organization
JavaCourses

If you have a calendar application that can export entries in vCalendar format, copy one entry to `toolkit/appdb/DefaultColorPhone/pim/events/Eevnts` . Otherwise, use the following sample vCalendar item, saving it as `sample.vcs`:

```
BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
DTSTART:19980114T210000Z
DTEND:19980114T230000Z
LOCATION:My office
CATEGORIES:Business
DESCRIPTION;ENCODING=QUOTED-PRINTABLE:This is a note associated with the meeting=0D=0A
SUMMARY:Meeting to discuss new architecture
PRIORITY:3
END:VEVENT
END:VCALENDAR
```

Now select **Event Lists** from the options shown in Figure 1. The resulting display will resemble Figure 5.

Figure 5: Access to vCalendar Information



Summary
Meeting to discuss new architecture

Note
This is a note associated with the meeting

End
11:00:00 PM
Wed, 14 Jan 1998

Start
09:00:00 PM
Wed, 14 Jan 1998

UID
sample.vcs

Location
My office

Conclusion

JSR 75, Optional Packages for the J2ME Platform, defines two optional packages: Personal Information Management and FileConnection. APIs in the PIM package give J2ME devices ready access to personal data of three popular types. This article surveyed the PIM APIs and presented a tutorial on their use. The sample code in this article showed how easy it is to develop MIDlets that read and write native contact, event, and to-do information. With only a handful of classes and interfaces you can begin developing MIDlets that process personal information on devices that support the PIM APIs optional package.

For more information

JSR 75: Optional Packages for the J2ME Platform
J2ME Wireless Toolkit 2.2
IBM's JSR 75 Reference Implementation
Nokia 6630
Internet Mail Consortium vCard and vCalendar Specifications

Acknowledgments

Special thanks to Stuart Marks of Sun Microsystems, whose feedback helped me improve this article.

About the Author

Qusay H. Mahmoud provides Java technology consulting and training services. He has published dozens of Java articles, and is the author of *Distributed Programming with Java* (Manning Publications, 1999) and *Learning Wireless Java* (O'Reilly, 2002).

Rate and Review
Tell us what you think of the content of this page.
○ **Excellent** ○ **Good** ○ **Fair** ○ **Poor**
Comments:

Your email address (no reply is possible without an address):
Sun Privacy Policy

Note: We are not able to respond to all submitted comments.
Submit »

E-mail this page Printer View

ORACLE CLOUD
Learn About Oracle Cloud
Get a Free Trial
Learn About PaaS
Learn About SaaS
Learn About IaaS

JAVA
Learn About Java
Download Java for Consumers
Download Java for Developers
Java Resources for Developers
Java Cloud Service
Java Magazine

CUSTOMERS AND EVENTS
Explore and Read Customer Stories
All Oracle Events
Oracle OpenWorld
JavaOne

COMMUNITIES
Blogs
Discussion Forums
Wikis
Oracle ACEs
User Groups
Social Media Channels

SERVICES AND STORE
Log In to My Oracle Support
Training and Certification
Become a Partner
Find a Partner Solution
Purchase from the Oracle Store

CONTACT AND CHAT
Phone: +1.800.633.0738
Global Contacts
Oracle Support
Partner Support

Hardware and Software, Engineered to Work Together

Subscribe | Careers | Contact Us | Site Maps | Legal Notices | Terms of Use | Privacy | Cookie Preferences |