



Using PIM API to Import/Export vCards

October 26, 2006

Technical Article

Using PIM API to Import/Export vCards

By
MOTODEV Staff

This article is an introduction about how to import vCard to PIM contact and export PIM contact to vCard. It includes a brief introduction of PIM API and vCard with a small example.

Overview of PIM and vCard

The PIM (Personal Information Management) optional package is one part of the JSR 75 PDA (The other is File Connection API). The PIM API offers programming interfaces for a handset's internal personal information database, which could be used to manipulate the personal information that includes the contact list, the ToDo list, and the Event list. The Motorola A1200 and ROKR E2 support JSR75 PIM in Motorola Linux platforms.

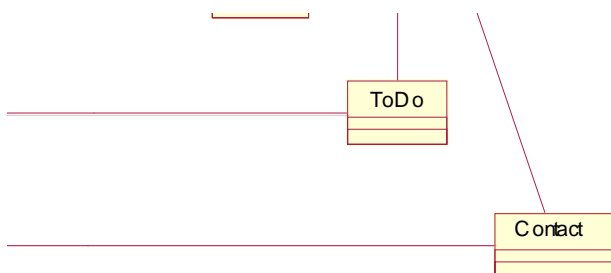


Figure 1: Class diagram of PIM package

VCard, also called electronic business card, is an industry standard format used to exchange contact information. The PIM API provides support for importing and exporting personal data in vCard and vCalendar formats.

A vCard data stream could include multiple vCard objects. A vCard object starts with a "BEGIN:VCARD" label and ends with an "END:VCARD" label. The body of a vCard object is a set of property-value pairs like:

```
Property Name [; Property Parameter]: Value
```

VCard item example:

```
TEL;HOME:861012345678
```

VCard example:

```
BEGIN:VCARD
VERSION:2.1
LABEL:BeiJing
FN:Test
TEL:1234567
END:VCARD
```

VCalendar, which includes vTodo and vEvent, has a similar structure like vCard. In this article, we just focus on Contact and vCard. ToDo, Event and vCalendar can be treated in a similar way.

VCalendar example:

```
BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
...
END:VEVENT
BEGIN:VTODO
...
END:VTODO
END:VCALENDAR
```

The Motorola A1200 and ROKR E2 support vCard 2.1 and vCalendar 1.0 in PIM package, and the default Charset is UTF-8.

PIM VCard example application

The example is easy to use: first, use the “Create Contact” command to create a Contact object; then use “Export VCard” to export the Contact object as a vCard to a RecordStore; next select the “Clear Contact” command to clear the text field and the contact entry in phonebook; finally, use the “Import VCard” command to get the vCard object from the RecordStore, show its attribute in the text field and put it in the phonebook.

The example also has a command called “Show Info” which shows the PIM package’s vCard and vCalendar version and prints the vCard content in the RecordStore.

Get PIM instance and contact list.

```
pim = PIM.getInstance();
try{
    contactList = (ContactList)pim.openPIMList(PIM.CONTACT_LIST, PIM.READ_WRITE);
}
catch(Exception e){
    ...;
}
```

Create a Contact object and put it in the device phonebook.

```
try{
    contact = contactList.createContact();
    if (contactList.isSupportedField(Contact.NAME))
        contact.addStringArray(Contact.NAME, PIMItem.ATTR_NONE, name);
    if (contactList.isSupportedField(Contact.TEL))
        contact.addString(Contact.TEL, PIMItem.ATTR_NONE, phoneNum);
    if (contactList.isSupportedField(Contact.ORG))
        contact.addString(Contact.ORG, PIMItem.ATTR_NONE, org);
    contact.commit();
}
catch(Exception e){
    ...;
}
```

Export Contact to vCard in RecordStore.

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
DataOutputStream os = new DataOutputStream(baos);
String[] data_formats = pim.supportedSerialFormats(PIM.CONTACT_LIST);
try {
    pim.toSerialFormat(c, os, "UTF-8", data_formats[0]);
}
catch (Exception pe){
    return;
}
byte[] vcContent = baos.toByteArray();
vcStore.setVCardRecord(vcContent);
```

Clear the text field and delete contact in phone book.

```
givenName.setString("");
familyName.setString("");
contactOrganization.setString("");
phoneNumber.setString("");
try{
    if(contact!=null){
        contactList.removeContact(contact);
    }
}catch(Exception e){
    ...;
}
```

Import the vCard

```
byte[] vcContent = vcStore.getVCardRecrod();
ByteArrayInputStream bis = new ByteArrayInputStream(vcContent);
DataInputStream is = new DataInputStream(bis);
try {
    PIMItem[] items = pim.fromSerialFormat(is, "UTF-8");
    Contact ct = (Contact) (items[0]);
```

```

Contact contact = contactList.importContact(ct);
if (contactList.isSupportedField(Contact.NAME)){
    name = contact.getStringArray(Contact.NAME, 0);
        givenName.setString(name[contact.NAME_GIVEN]);
        familyName.setString(name[Contact.NAME_FAMILY]);
    }
    if (contactList.isSupportedField(Contact.TEL)){
        String phoneNum = contact.getString(Contact.TEL, 0);
        phoneNumber.setString(phoneNum);
    }
    if (contactList.isSupportedField(Contact.ORG)){
        String org = contact.getString(Contact.ORG, 0);
        contactOrganization.setString(org);
    }
    contact.commit();
}
catch(Exception e) {
    ...;
    return;
}

```

The PIM package is protected by the Java™ ME security mechanism, and the PIM related permissions:

```

javax.microedition.pim.ContactList.read
javax.microedition.pim.ContactList.write
javax.microedition.pim.EventList.read
javax.microedition.pim.EventList.write
javax.microedition.pim.ToDoList.read
javax.microedition.pim.ToDoList.write

```

MIDlet permissions for this example:

```

MIDlet-Permissions: javax.microedition.pim.ContactList.read,
javax.microedition.pim.ContactList.write

```

Conclusion

With the JSR75 PIM API, a Java™ ME application can communicate with external platforms via a well-known industry standard. For example, the handset could send the contact list in vCard format to a PC with Bluetooth allowing Microsoft Outlook® to open it directly and import it to the address book. Using the vCard import/export function, we will be able to build a power personal information management and synchronization Java™ ME application.

References

JSR-75: PDA Optional Packages, <http://jcp.org/aboutJava/communityprocess/final/jsr075/index.html>

vCard 2.1 <http://www.imc.org/pdi/vcard-21.txt>

vCalendar 1.0 <http://www.imc.org/pdi/vcal-10.txt>