The JSR 281 IMS Services API: Time to Deliver

Stefan Svenberg and Niclas Palm

IMS Java Standardisation
Ericsson AB
http://www.ericsson.com

TS-5102

Goal of This Talk

What you will remember

In only three basic steps, you will learn how to write a Java™ Platform MIDlet using the Java Specification Request (JSR) 281 API to start sharing multimedia content

Agenda

Why we are here

IP Multimedia Subsystem (IMS)

IMS and JSR 281 IMS Services API

Develop a chat MIDlet for text and pictures

Improve with sending a custom ring signal

Improve with video

Agenda

Why we are here

IP Multimedia Subsystem (IMS)

IMS and JSR 281 IMS Services API

Develop a chat MIDlet for text and pictures

Improve with sending a custom ring signal

Improve with video

Why We Are Here

- Last year at The 2006 JavaOne℠ Conference in session TS-3319 “PoC and Beyond: The JSR 281 IMS Services API”, the speakers claimed (quote):

  “It [the JSR 281] also draws a picture of the **cool Java applications** becoming possible with this API and how they can enrich the mobile Java technology world.”

- In this session, a **cool Java platform IMS application** will be described and demonstrated

Agenda

Why we are here

IP Multimedia Subsystem (IMS)

IMS and JSR 281 IMS Services API

Develop a chat MIDlet for text and pictures

Improve with sending a custom ring signal

Improve with video

IMS: The IP Multimedia Subsystem

What it is

- Purpose: Move **all** voice and multimedia communication to packed-based technologies, to merge telephony (fixed and mobile) and Internet
- Architecture: A standardized core IP network infrastructure that serves as a **common foundation** for higher-level services
- Operators: Offer new multimedia services rapidly at **lower cost**
- Users: **Peer-to-peer multimedia real-time communications** in a highly controlled and personalized way

Example IMS Use Cases

Business case easier if made specific

- Instantly share a camera snapshot with a friend
- “You see what I see”—stream live video to your friend’s phone or computer
- Multi-media telephony
- Exchange multimedia with whom you are talking to on a phone
- Networked multi-user gaming
- Multi-user Push-to-talk over Cellular (PoC)
- White-boarding
- And more

Agenda

Why we are here

IP Multimedia Subsystem (IMS)

IMS and JSR 281 IMS Services API

Develop a chat MIDlet for text and pictures

Improve with sending a custom ring signal

Improve with video

JSR 281 Role in IMS

Overview

- IMS does not standardize applications, or depend on an application framework
- Java provides a widespread framework where portable applications can be built and deployed
- The JSR 281 standardizes an API to enable a MIDlet to:
  - Act as a client to IMS application servers
  - Realize a peer-to-peer service on a mobile device and expose it to the IMS
- Support for more IMS functionality can be added to the JSR 281 (future proof)
- A Java platform developer does not have to know IMS

JSR 281 Overview

Feature Highlights

- An API to the IMS functionality of a mobile device:
  - Core API: To access **service independent** IMS primitives; this one is mandatory
  - Service API: To access **standardized** service enablers; these are all optional
- Targets Java platform profiles based on **CLDC**/CDC
- **High-level abstraction** hides the IMS technology
- Allows **low-level access** for advanced usage
- **Extensible** to make it future-proof

Core API

Package highlights

- Javax.microedition.ims
  - Entry point to IMS functionality
  - Select type of services
- Javax.microedition.ims.core
  - Create IMS calls (“sessions”) to a remote peer
  - Communicate with IMS application servers
  - Query capabilities of a remote peer
  - Send references
- Javax.microedition.ims.core.media
  - Define the media objects that carry content flows
  - Set up media players for streaming media
  - Quality of service

JSR 281 IMS Services API

Package highlights—OMA enablers

- Javax.microedition.ims.poc
- Javax.microedition.ims.presence
- Javax.microedition.ims.xdm

JSR 281 Status and Timeline

- EDR (Early Draft Review)—Q4 2006
- PDR (Public Draft Review)—Q2 2007
- PFD (Proposed Final Draft)—Q3 2007
- FR (Final Release)—Q4 2007

Agenda

Why we are here

IP Multimedia Subsystem (IMS)

IMS and JSR 281 IMS Services API

Develop a chat MIDlet for text and pictures

Improve with sending a custom ring signal

Improve with video

JSR 281 Usage

Principal steps

- A JSR 281 IMS application goes through three **generic** steps:
  - Access the IMS functionality of the device
  - Go online to the IMS network using a selected identity
  - Connect a call including media flows with a remote IMS device
- The precise details at any step depends on the intrinsic application logic, and what type of service it realizes

Example: A Cool IMS Java Application

Getting concrete

- We use an application example to probe further into the Core API of the JSR
- Basic example: An simple IMS Java platform chat MIDlet to exchange text and pictures
- Developed and installed to Alice's and Bob's mobile devices
- Alice wants to chat with Bob
- After that, a set of improvements are applied

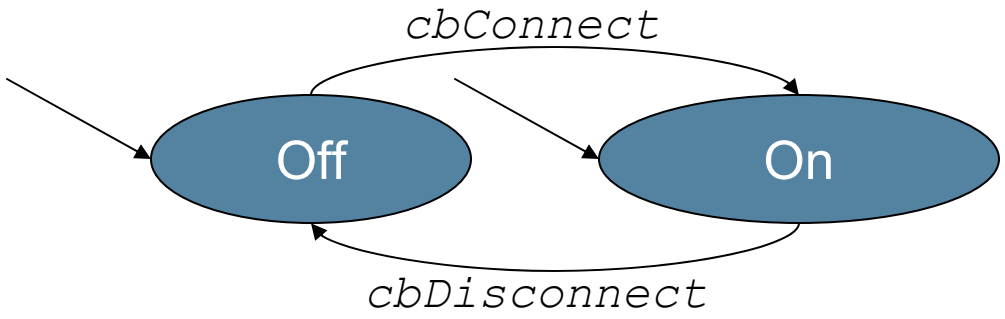
JSR 281 Manager

Access the IMS functionality using the ImsManager

- The Manager object represents the complete IMS functionality of the local device specifically for use by the application
- The Manager has:
  - A state machine for IMS connection
  - Method to get a list of user identities
  - A factory function for Service creation (next step)
  - A registry that records properties of the application
- A listener for changes in IMS connection state

JSR 281 Manager

Manager state machine



Legend:
- Arc label in *italics* is an IMS state listener call back when IMS connection state is changed
- State label in **bold** and solid circle is a steady state

Manager Pseudo Code

```
{
    // Application gets the manager under its name
    mgr = Manager.getManager("com.j1.chatMidlet");
    // Get the list of IMS provisioned user id's
    if (mgr.getState() == IMS_STATE_ON)
    {
        userIds = mgr.getUserIdentities();
        // app may present these to the user to select one
        userId = selectId(userIds);
    }
}

// next: create service
```

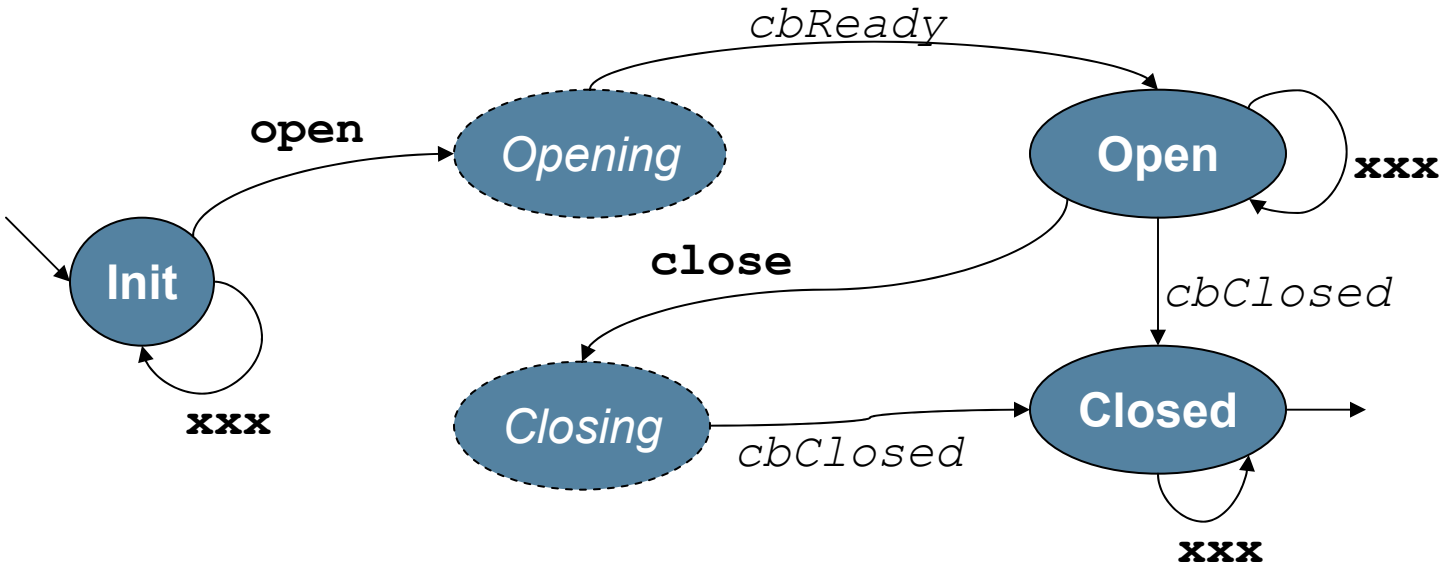
JSR 281 Service

Go online using the Service object

- The application creates a Service object when it is prepared to handle calls
- The Service interface is service-independent, and has
  - A state machine
  - Some methods to trigger state transitions
- Extensions of the Service interface represent services for core and the supported enablers
- A listener on Service for state events

JSR 281 Service

Service object state machine (simplified)



Legend:
- Arc label in **bold** face is a Service interface method that cause state change
- Arc label in *italics* is a listener call back when the state machine has changed state due to some IMS event
- State label in **bold** and solid circle is a steady state
- State label in *italics* and dashed circle is a temporary state expecting an event
- Arc label with **xxx** is a placeholder for service-specific methods and callbacks in that state

JSR 281 Core Service

Go online using the Service object

- The Core service is used to represent the core functionality (media sharing)
- The state machine semantics
  - Init state: The service is not yet online; the user identity is set here
  - Open state: The service is online and functional; sessions can be created here, both incoming and outgoing
  - Closed state: The service has been de-activated
- Core service listener for Core-specific service events

Service Pseudo Code

```
{
    // Core Service creation
    srv = (CoreService)mgr.createService("CoreService");
    // srv is now in the init state.
    // Downcasted, we are allowed to set the user id.
    srv.setUserId(userId);
    // Go online!
    srv.open();
}
// call back generated when network accepts
// or rejects
// next: create the session
```

JSR 281 Session

Make IMS calls

- The Session object represents a call between the end points
- The remote end point is addressed via the remote user id
- The session object has:
  - A state machine
  - Media objects
- A listener on session events

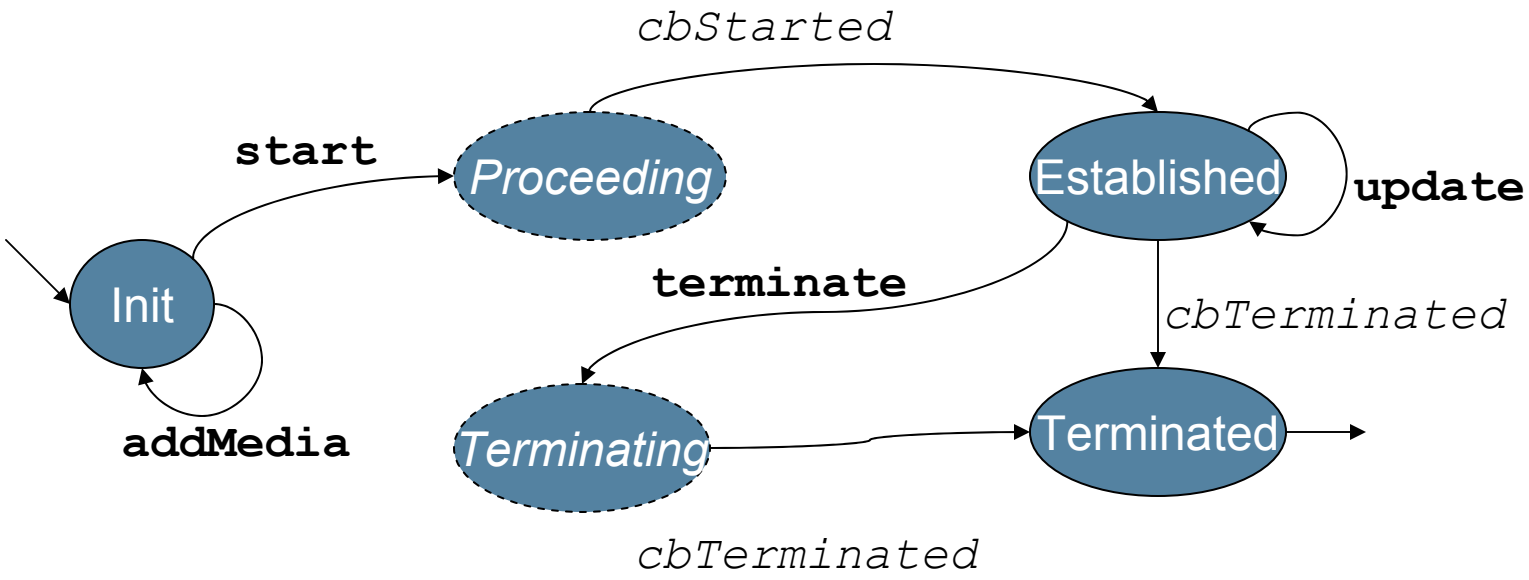
JSR 281 Session

State machine

- The session state machine:
  - Init state: the caller adds media to the session
  - Proceeding state: the callee receives the session
  - Established state: the IMS call has been accepted at the remote, quality of service is set, and media content starts flowing
  - Terminating state: the IMS call has ended

Session

Session object state machine (simplified)



Legend:
- Arc label in **bold** face is a Session interface method that cause state change
- Arc label in *italics* is a listener call back when the machine has changed state due to some IMS event
- State label in **bold** and solid circle is a steady state
- State label in *italics* and dashed circle is a temporary state expecting an IMS event

Session Pseudo-Code for Alice

```
{
    // Alice to select the remote user id of Bob's
    remoteUserId = selectRemoteUserId();
    ses = srv.createSession(remoteUserId);
    // Set the content types and bi-directional session
    String[] acceptedTypes =
        new String[] {"text/plain", "image/jpeg"};
    med = ses.createFramedMedia(acceptedTypes, SEND_RECV);
    ses.start();
    ...
    // after Bob's accept, Alice texts him
    msgId = med.sendBytes("Hello Bob!", "text/plain");
    ...
}
// next: Alice shares a camera snapshot instantly
```

Picture Send Pseudo-Code for Alice

```
{
    // Alice to use the JSR 135 for camera
    Player p = Manager.createPlayer("capture://video");
    p.realize();
    VideoControl vc = p.getControl("VideoControl");
    vc.initDisplayMode(USE_DIRECT_VIDEO, canvas);
    vc.setVisible(true);
    p.start();
    // Take a snapshot as jpg and send it
    byte[] buf = vc.getSnapshot("image/jpeg");
    msgId = myMedia.sendBytes(buf, "image/jpeg);
}
// next: Bob to receive messages
```

Session Pseudo-Code for Bob

```
{
    // Bob to accept session invite
    ses.accept();
    ...
    // Bob receives picture and text
    if ("image/jpeg".equals(contentType)) {
        media.receiveFile(messageId, "image.jpeg");
        // Display
    }
    else if ("text/plain".equals(contentType)) {
        messText = media.receiveBytes(messageId);
        // print the text
    }
}
```

DEMO

Running the chat MIDlet

Agenda

Why we are here

IP Multimedia Subsystem (IMS)

IMS and JSR 281 IMS Services API

Develop a chat MIDlet for text and pictures

Improve with sending a custom ring signal

Improve with video

Call With a Ring Signal

- A ring signal consists (here) of:
  - A picture
  - A ring tone
- The caller sets this when making the call; the callee renders when it is alerting
- The application-specific solution uses an interface to the underlying SIP protocol
- Demonstrates **advanced low-level access** to SIP headers and body of the INVITE message
- Caution: Interoperability!

Send Ring Signal for Alice

```
{
    // Alice created her session object as before
    ses.addHeader("P-ImageIncl", "yes");
    MessageBodyPart imgPart = ses.createBodyPart();
    imgPart.setType("image/jpeg");
    imgPart.setContent(imageData);

    ses.addHeader("P-RingtoneIncl", "yes");
    MessageBodyPart melPart = ses.createBodyPart();
    melPart.setType("audio/midi");
    melPart.setContent(ringtoneData);

    ses.start();
}
// next: Bob to receive
```

Receive Ring Signal for Bob

```
{
    // Bob accepts, and gets the image and ring tone
    Message mess = ses.getRequest(METHOD_INVITE);
    if ("yes".equals(mess.getHeader("P-ImageIncl"))) {
        // [code to loop through headers omitted]
        if ("image/jpeg".equals(parts[i].getType())) {
            imageData = parts[i].getContent();
            // Bob displays it
        }
    }
    if ("yes".equals(mess.getHeader("P-RingtoneIncl"))) {
        // [code to loop through body parts omitted]
        if ("audio/midi".equals(parts[i].getType())) {
            ringtoneData = parts[i].getContent();
            // Bob plays it
        }
    }
}
```

DEMO

Ring Signals

Agenda

Why we are here

IP Multimedia Subsystem (IMS)

IMS and JSR 281 IMS Services API

Develop a chat MIDlet for text and pictures

Improve with sending a custom ring signal

Improve with video

Include Streaming Video

- IMS end to end is often illustrated with sharing of live streaming video
- Knowing Bob likes surfing, Alice wants to stream live video of the waves to her friend while chatting at the beach
- This example shows adding a streaming media object to the session with playback

Send Video for Alice

```
{
    // Session created as before
    StreamMedia myMedia = ses.createStreamMedia("video",
"myClip.3gp");
    mySession.start();

    // Alice can see the outgoing video
    ImsPlayer p = myMedia.getSendingPlayer();
    p.start();
}
```

Receive Video for Bob

```
{
    // Session created as before
    Vector media = session.getMedia();
    StreamMedia myMedia = media.firstElement();
    ImsPlayer p = myMedia.getReceivingPlayer();
    p.start();
}
```

DEMO

Video

More Functionality in the Core API

- Audio streaming of speech or music
- Application-specific media
- Query capability of the remote
- Use of application servers on the network
- Session updates
- Listener methods
- Media control
- Java Application Descriptor (JAD) file properties

Summary

- Uses for the JSR 281 IMS Services API
- Role of Manager, Service, and Session objects
- Chat text MIDlet developed
- Improving for picture attachments
- Added real-time streaming video
- No detailed IMS knowledge needed

For More Information

See also:

- Session TS-3319— “PoC and Beyond: The JSR 281 IMS Services API” (The JavaOne^{SM} 2006 Conference technical session)
- http://www.jcp.org/en/jsr/detail?id=281
- 3GPP TS 23.228, 3GPP TS 24.229
- IETF RFC 3261 SIP, IETF RFC 2327 SDP
- OMA: http://www.openmobilealliance.org
- The 3G IP Multimedia Subsystem, 2nd edition, G. Camarillo and M. Garcia-Martin, Wiley, 2006
- The IMS IP Multimedia Concepts and Services, 2nd edition, M. Poikselkä and Co., Wiley, 2006
- Booth 1118—Mobility Village

Q&A

Stefan Svenberg

Niclas Palm

The JSR 281 IMS Services API: Time to Deliver

Stefan Svenberg and Niclas Palm

IMS Java Standardisation
Ericsson AB
http://www.ericsson.com

TS-5102