Mobility Service Oriented Architecture Extending SOA to Mobile Devices

Ari Shapiro/Andreas Frank

Engagement Architects

Sun Microsystems, Inc.
http://www.sun.com

TS-5639

Creating Network-Based Mobility Services

Learn how to architect, build, and deploy dynamic network-based Java™ platform mobility services

Agenda

Why Do We Need MSOA?

What Is MSOA?

Demo—Services and Event Manager

Server Platform

Client Platform

Demo—MSOA Use Cases

Q&A ~ 5 Minutes

Why Do We Need MSOA?

- It is challenging to create and roll out new dynamic network-based Java platform services especially to existing devices
- Typical roll-out time for new service is currently measured in several months to more than a year
- Exposing existing network services/infrastructure to mobile clients is difficult
- Rolling out IMS services to existing non-IMS clients (mobile devices, STBs, etc.) is a major challenge

Customer Pain Points

- Faster time to market
- Revenue growth
- Composed and blended services
- “Over-the-top” services (IP-based)
- Software-as-a-Service (SaaS)
- Cross-access network personalization
- Interactive and context aware Services
- Access and provisioning across converged networks and federated service domains
- Synchronous and asynchronous

Agenda

Why Do We Need MSOA

What Is MSOA?

Demo (Services and Event Manager)

Server Platform

Client Platform

Demo—MSOA Use Cases

Q&A

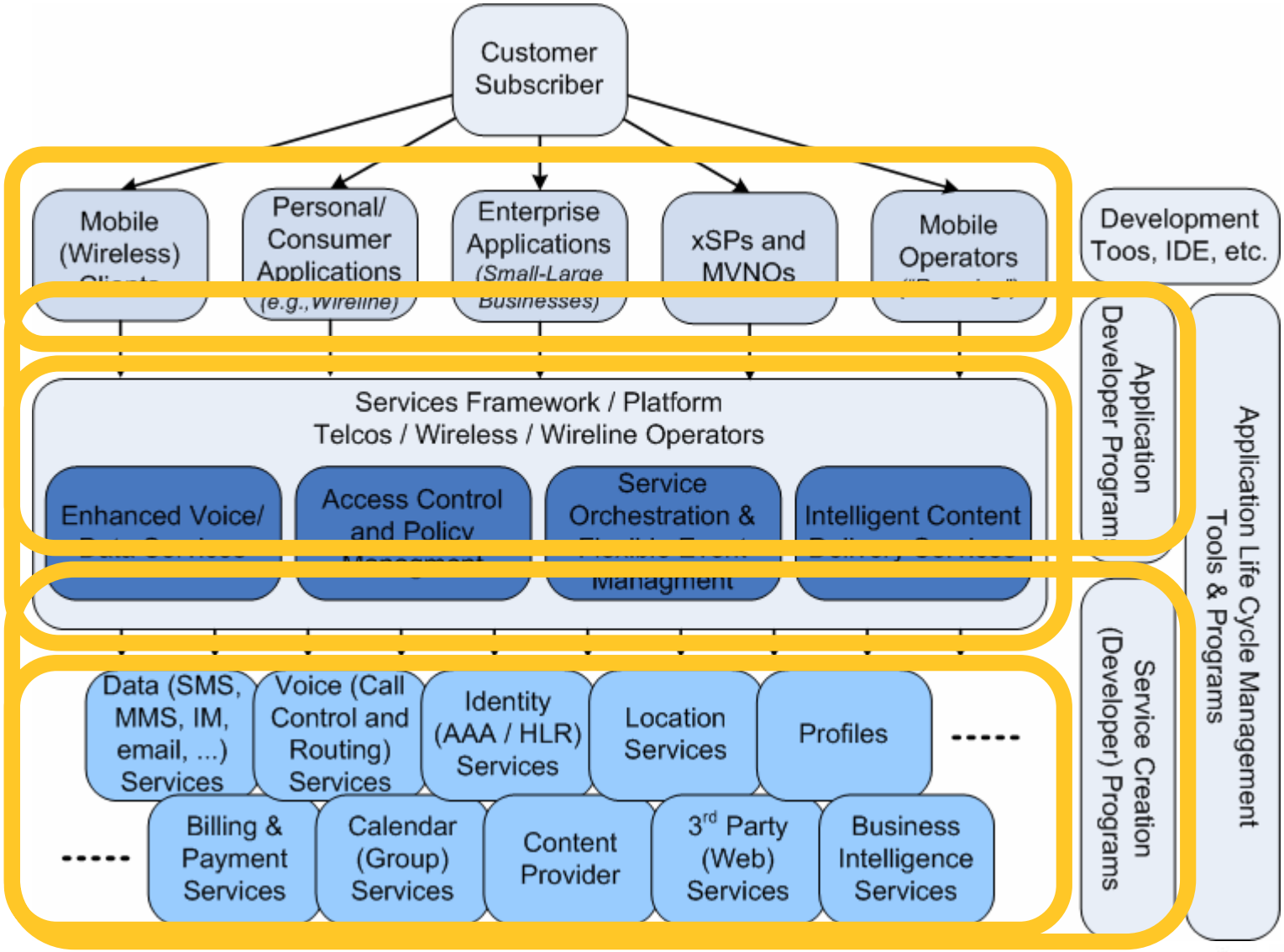
What Is MSOA?

- A **M**obility **S**ervice **O**riented **A**rchitecture framework and platform that leverages existing and emerging **industry standards and technologies** and combines them through a **best practice** approach
- Extends SOA to mobile devices, STBs, sensors,…
- Provides the framework for customers to **rapidly create** and **deploy end-to-end Java platform services**— from the client (e.g., Midlet, Xlet) application through a collaboration of back-end web services and/orIMS services
- A full client side framework and set of management applications

What Is MSOA?

- Server platform
  - Uses Sun's Solaris™ Operating System (Solaris OS), Java Enterprise Systems (Java ES), and tools
  - Enables the easy creation and combination of services
- Client platform
  - Provides a forward looking consistent client platform that can run on today's devices
  - Allows to leverage high-end device functionality

MSOA—High Level Overview



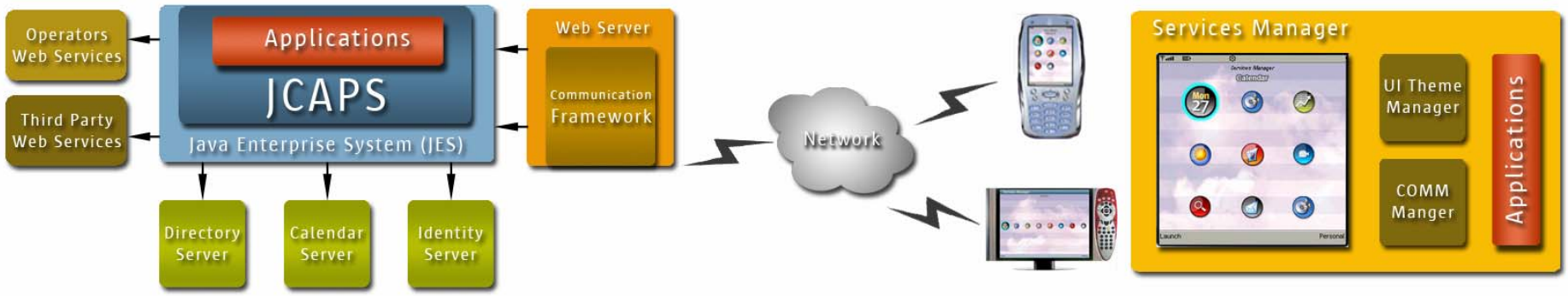
MSOA Benefits

- Add/remove/configure services on the fly
- Identity-based network services
- Faster time to market
- Integrates existing services as well as new IMS services easily
- Little or no dependency on device features
- Supported by a suite of development tools which make development easier and faster

MSOA Benefits

- Follow principles and practices for designing shared, reusable, distributed services
  - Much easier to pull off with a consolidated stack
- SOA value
  - Separation of service interface from underlying implementation (loose coupling)
  - Promotes service reuse through discoverable and self-describing services
  - Services are course-grained, compose-able, and rely on a standards-based infrastructure

MSOA High-Level Components



Agenda

Why Do We Need MSOA

What Is MSOA?

Demo—Services and Event Manager

Server Platform

Client Platform

Demo—MSOA Use Cases

Q&A

DEMO

Services Manager
Event Manager
Provisioning Screens

Agenda

Business case for MSOA

What Is MSOA?

Demo—Services and Event Manager

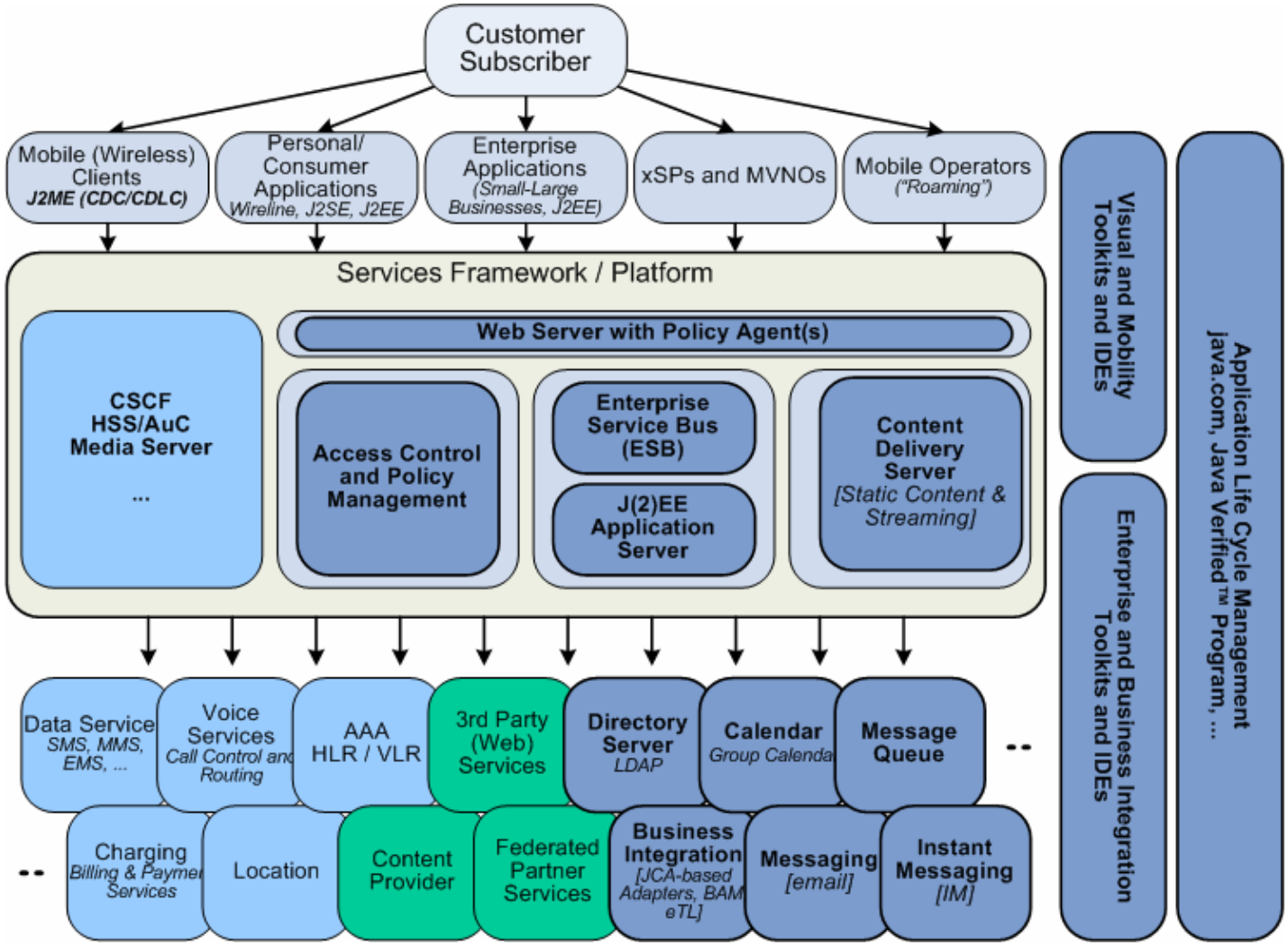
Server Platform

Client Platform

Demo—MSOA Use Cases

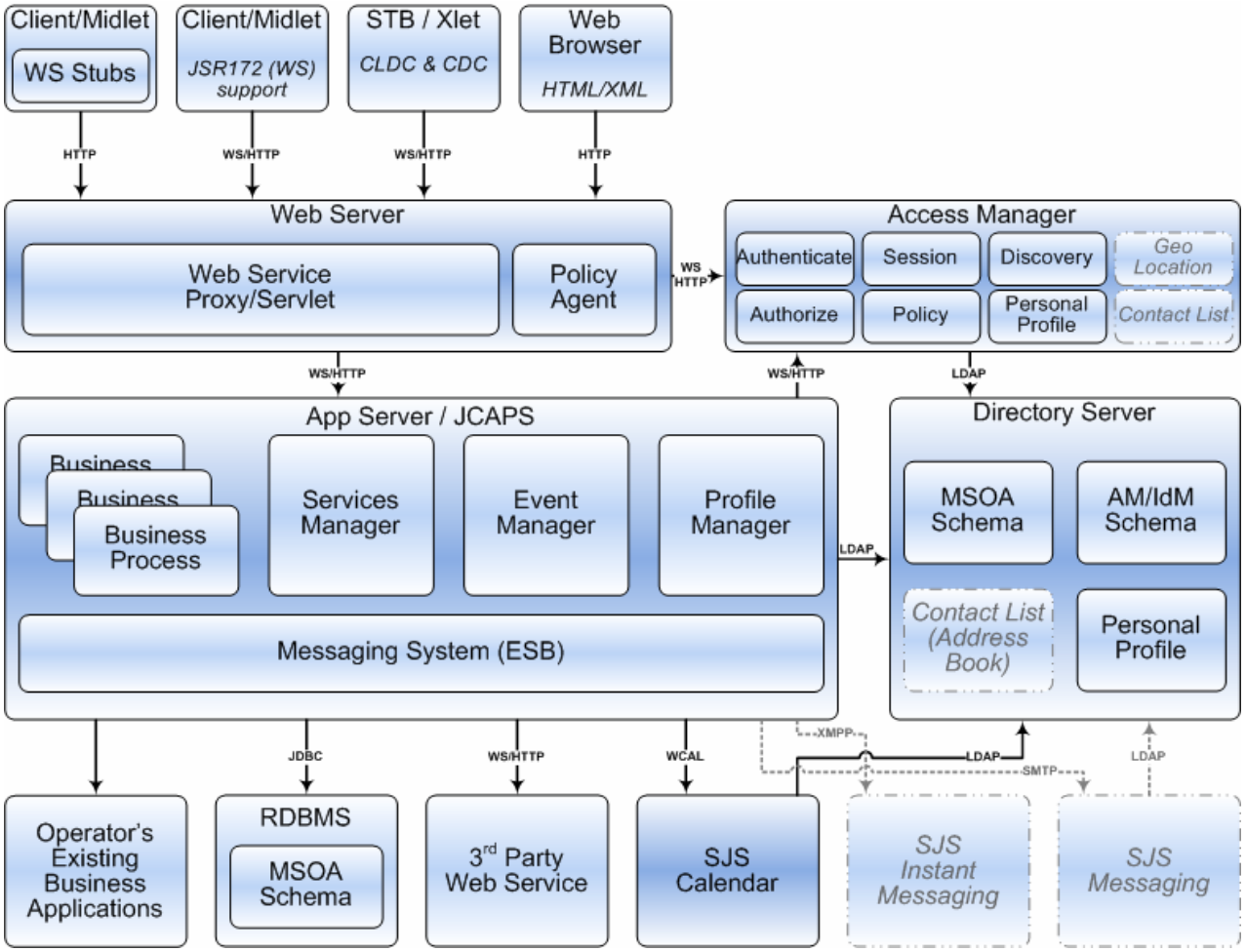
Q&A

MSOA—Generic Components



J2ME = Java 2 Platform, Micro Edition (J2ME™ platform); J2SE = Java 2 Platform, Standard Edition (J2SE™ platform).

Software Components and Interfaces



Agenda

Business Case for MSOA

What Is MSOA?

Demo—Services and Event Manager

Server Platform

Client Platform

Demo—MSOA Use Cases

Q&A

Client Platform

Today MIDP 2.0-based, easily extensible to CDC

- Services Manager MIDlet
  - Provides one point of entry for all services
  - Services are received from the server and can be dynamically changed (updated/added/removed separately)
- Event Manager MIDlet
  - Provides a single point of entry for all events received by the client
- UI Manager
  - Pure Java technology, will run on any MIDP 2.0 device
  - Consistent look and feel across multiple devices
  - Theme manager

Client Platform

Today MIDP 2.0-based, easily extensible to CDC

- Communications Manager
  - HTTP-based communications manager using an optimized binary protocol
- Event Manager
  - Provides a single point of entry for all events received by the client
- Development Tools—Based on Netbeans™ Software Mobile and the Sun Java Wireless Toolkit

Communication Manager

- Http-based communication with the server
- Uses an optimized binary protocol to transfer Objects
- Supported by Netbeans Software Mobile
- Supports access to web services without needing Java Specification Request (JSR) 172 on the device

UI Manager

- MIDP 2.0 Pure Java technology-based
- Runs on today's devices
- Supports application theming
- Provides a number of layout managers (Box, Grid, Flow) out of the box

Client-Side Challenges

Major client-side issues

- Launch MIDlet from another MIDlet
  - The Services Manager needs to be able to launch other services which are implemented as MIDlets
  - Every MIDlet needs to be able to go back to it's caller
- Client-side caching
  - Data needs to be cached on the client easily by applications
- Cross-device UI
  - The MSOA UI Manager provides a framework to build UIs that will work on multiple devices
- CDC and CLDC support

Launching a MIDlet From Another MIDlet

Deep dive into the problem

- Available options
  - JSR 211 CHAPI
    - Pros—Standards-based
    - Cons—Very few devices today support JSR 211
  - Proprietary APIs (e.g., Muglet API)
    - Pros—Deployed on a lot of devices
    - Cons—Not standards-based
  - Platform request
    - Cons—Implementation dependent
- MSOA framework handles this transparently for you
- Supports MVM as well

Using JSR 211 CHAPI

- JSR 211 CHAPI allows an application to register itself as a handler
- Every MSOA MIDlet registers as a handler
  - MicroEdition-Handler-1-ID = fully qualified name
- The Services Manager uses the fully qualified name it receives from the server to launch the service
- When the called service exits control reverts back to the launching MIDlet

Proprietary APIs

Muglet API example

- The Muglet API provides a facility to launch a MIDlet from another MIDlet
- The following code launches a MIDlet:

```
// launch the specified midlet
com.sprintpcs.util.System.setExitURI(midlet:fullQualifiedName?params);
// the above specified MIDlet will only launch after
// the current MIDlet exits
launchingMidlet.notifyDestroyed();
```

Using Platform Request

- On some devices Platform Request can be used to launch a MIDlet from another MIDlet
- The following code launches another MIDlet

```
// launch the specified midlet, needToExit let's us know
// if we need to exit the current MIDlet
boolean needToExit =
  launchingMidlet.platformRequest("midlet:fullyQualified
  Name”);
// if we need to exit the current MIDlet
if (needToExit) {
   launchingMidlet.notifyDestroyed();
}
```

MSOA MIDlet Launch Framework

- Uses Factory pattern to create a launcher
- All launchers implement the MidletHelper interface
- MidletHelper provides the interface to launch a MIDlet and go back to the caller

MSOA MidletLauncherFactory

Factory class to create the appropriate launcher

```
// first see if the platform supports Muglets
    try {
        Class.forName("com.sprintpcs.util.Muglet");
        return new MugletMidletHelper();
    } catch (ClassNotFoundException ex) {
         // next try CHAPI
          try {
       Class.forName("javax.microedition.content.Registry");
             return new ChapiMidletHelper();
       } catch (ClassNotFoundException e) {
             //none of the above use Platform Request
             return new PlatformRequestMidletHelper();
       }
    }
```

Using the MSOA MIDlet Launch Framework

- Using this framework from a client is really simple
- Below is code taken from the Services Manager

```
// launch the selected MIDlet
protected void launchMidlet(int midletNum) {

  helper.launchMidlet(managerMidlet,midlets[midletNum].g
  etPackagename());
}
```

Agenda

Why Do We Need MSOA?

What Is MSOA?

Demo—Services and Event Manager

Server Platform

Client Platform

Demo—MSOA Use Cases

Q&A

DEMO

The mSOA Framework in Action
- Dynamic Deployment of Services
- Conference Call Setup
- Location-Based Services

Summary

- Extends SOA to mobile devices, STBs, sensors…
- Provides the framework for customers to **rapidly create** and **deploy end-to-end Java platform services**
- Leverages existing and emerging **industry standards** **and** **technologies** and combines them through a **best practice** approach

Q&A

Ari Shapiro/Andreas Frank

Mobility Service Oriented Architecture Extending SOA to Mobile Devices

Ari Shapiro/Andreas Frank

Engagement Architects

Sun Microsystems, Inc.
http://www.sun.com

TS-5639

Back-Up Slides

Cross Device UI

- The MSOA UI Manager provides a framework to build UIs that will work on multiple devices

Services Manager MIDle

- The Services Manager serves as a mini-AMS
- It displays the services the user has
- Each service is a separate MIDlet
- The user can launch his services directly from the Services Manager

Event Manager MIDlet

- The Event Manager serves as one point of entry for all events
- Events responses are context sensitive
- The user can launch a response to an event right from the Event Manager

MidletHelper Interface Methods

```
// launch the specified midlet with parameters
 public void launchMidlet(MIDlet launchingMidlet,
   String newMidletName, String[] params);
// launch the specified midlet no params
 public void launchMidlet(MIDlet launchingMidlet,
   String newMidletName);
// go back to the MIDlet that called you
public void goBackToCallingMidlet(MIDlet
   launchingMidlet);
```

MSOA Caching Framework

- Data needs to be cached on the client to provide better performance and less network traffic
- MSOA provides a caching framework that can be used by all MSOA clients to cache data, preferences, etc.

MSOA Design Principles

- Whole services lifecycle including creation, deployment, delivery, management, and execution which entails a Services creation and execution environment
- Tool support
- New generation adopt SOA for telco/cable
  - Layering
  - Web services
  - Orchestration, etc.
- Identity plays a central role
  - Third-party services providers integration (Liberty)

Event Manager Server Access Workflow