OSGi Service Platform
– Open Services Gateway Initiative –

Ikuo YAMASAKI
yamasaki.ikuo@lab.ntt.co.jp

NTT Cyber Solutions Laboratories, NTT Corporation

September 28, 2005
Tutorial in APNOMS 2005, Okinawa, Japan

Outline

I. OSGi Alliance and its Background
II. Features of OSGi
III. Use Cases of OSGi
IV. Technical Details of OSGi FW
V. Standard Services in Spec. Release 3.0
VI. Management Issues for providers
VII. Summary
OSGi Alliance

- **OSGi: Open Services Gateway Initiative**
  - Open forum, established in March 1999.
- **Specify and promote adoption of**
  - An open service platform for the delivery and management of multiple applications and services, using networked devices.
- **Initial target**: Residential Internet gateways with home automation applications.

---

**Background of Home NW and Services**

- Diversity, Coexistence, Update / Change
Problems of Service Provision to Residential Users (1)

Fixed “Services & Home Gateway”
Difficult to attach new devices and to modify Services

Problems of Service Provision to Residential Users (2)

- From Service Providers’ point of view,
  - Need to develop complete unique system
    - For each: service menu
    - For each: new devices and protocols
  - Development costs excessive.

From Service Providers’ point of view,
- Need to develop complete unique system
  - For each: service menu
  - For each: new devices and protocols
- Development costs excessive.
Problems of Service Provision to Residential Users (3)

- From end users’ point of view,
  - Need to set
    - Multiple home gateways and
    - Multiple access points to control devices from outside,
  - For multiple services.

Requirements of Home Gateways

- To provide services via connected devices to residential users,
  - Collaboration and communication between multiple devices that use same protocol as well as different ones.
  - Support of new devices, new protocols, new services, and bug fix.
  - Use cases will be diverse. However, reduce the cost of system development.
    - Multipurpose and open platform.
    - Must NOT be influenced by specified business fields or companies.
Key Idea: Open Services Gateway functions

- Software component architecture.
  - Compose functionalities of networked devices by software components
- Dynamic delivery and update of software components via network for
  - Adapting various services, devices, protocols
  - Customization for gateways’ environment and users’ requirements
  - Bug fixes
- Bridge multiple networked devices speaking different protocols.
- Open platform.

OSGi Service Platform

- The latest specification is Release 3 (March, 2003)
  - Fully open to anyone.
- Currently Release 4 is under development.
  - Provided only to OSGi Alliance members.
- Java based software component architecture
  - Portable technology.
  - Runs on multiple platforms (Operation Systems, devices).
    - Residential gateways,
    - Embedded appliance,
    - Desktop PCs,
    - Industrial computers,
    - Enterprise servers,
    - Automotive,
    - Smart phones,
- Minimum Environment for JVM to support is defined by Release 3
  - Subset of J2ME (CDC 1.0 + Foundation Profile 1.0)
**OSGi Alliance Members**


44 members (as of end-July, 2005)

(*) The companies Board of Directors belong to

---

**Organization of OSGi Alliance**

- **President**
  (John Barr, Motorola)

- **Board of Directors**
  (13 directors)

- **Officers**
  (11 officers including President)

- **Expert Groups**
  (3 groups)
  - Core Platform Expert Group
  - Vehicle Expert Group
  - Mobile Expert Group

- **Committee Working Groups**
  (2 committees)
  - Marketing Working Committee
  - Market Requirement Working Committee
OSGi User Forums

- User Forum Japan
  - Established in Sept. 2004
    - 1st user group authorized by OSGi Alliance.
  - Members: 45 companies (as of end-July, 2005)
  - Charter
    - Promotion of OSGi service platform in Japan.
    - Exchanging/sharing views of experiences of OSGi.
    - Encouraging business collaborations.

- User Forum Korea
  - [http://www.osgikorea.org/](http://www.osgikorea.org/)
  - Established in May 2005

Outline

I. OSGi Alliance and its Background
II. Features of OSGi
III. Use Cases of OSGi
IV. Technical Details of OSGi FW
V. Standard Services in Spec. Release 3.0
VI. Management Issues for providers
VII. Summary
Features of OSGi

- OSGi defines spec of
  - OSGi Framework (FW)
    - Bundle Lifecycle Management
    - Package Sharing
    - Service Registry
  - Standard services
- Policy Free and Open Remote Management
- Separation of Spec and Implementations

Framework and Bundles

- OSGi Framework (FW); provides runtime environment of software components, called “Bundles”,
  - One of Java Applications
    - has a class of “static public void main(String args[])” method.
  - Run 1 FW on 1 JVM.
- Bundles; a unit of deliverable software component in OSGi.
  - Multiple bundles run on 1 FW.

In case of Java applications

<table>
<thead>
<tr>
<th>Java Appl1</th>
<th>Java Appl2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java VM</td>
<td>Java VM</td>
</tr>
<tr>
<td>OS</td>
<td></td>
</tr>
</tbody>
</table>

In case of OSGi bundles

<table>
<thead>
<tr>
<th>Bundle1</th>
<th>Bundle2</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSGi FW</td>
<td>OSGi FW</td>
</tr>
<tr>
<td>Java VM</td>
<td>Java VM</td>
</tr>
<tr>
<td>OS</td>
<td>OS</td>
</tr>
</tbody>
</table>
OSGi Framework (FW)

- OSGi Spec defines **FW mechanisms** for
  - managing *bundles* on a local *FW*.
  - *bundles’* collaborations on a local *FW*.

  1. Bundle Lifecycle Management
  2. Package Sharing
  3. Service Registry

**Bundle Lifecycle Management**

- MA (Management Agent) bundle can manage lifecycles of other bundles on the same FW.
  - Install / start / stop / uninstall / update.
  - MA (Management Agent): privileged bundle allowed to do critical operations, e.g., lifecycle management of other bundles, on a FW. Usually provided by the platform operator.

![Diagram of Bundle Lifecycle Management](image)
Java Package sharing between bundles

- **Java Package**: A group of Java classes. Constructed hierarchically.
- A bundle can export a package to FW and other bundles can import it and use its classes.
  - only one bundle is required to contain the package, and to load its classes into memory once.
- Reduce bundle size and memory consumption.
  - Especially effective for commonly used library packages.

Cooperation between bundles: Service

- Unit of functionality provided by a bundle and used by other bundles.
  - When MA installs and starts bundle A,
  - bundle A registers its service object into FW's Service Registry in order to provide functionality to other bundles.
  - Other bundle B and C can use this service, by invocation of its methods.
- Developers can define original services by themselves.
Structure of an OSGi Gateway

- Application Bundles (Developed for particular usage)
- Standard Services defined by OSGi Alliance
- FW registering Standard services
- Bundles registering Standard services

Standard Services in OSGi Release 3

- FW is designed to be lightweight.
- Other services are optional
  - Platform operators can choose the ones to deploy as needed or useful.
Example of service usage

- Printer control
- AV appliance control
- Car navigation control
- Log Service
- HTTP Service
- XML Parser Service
- UPnP Device Service

Control devices through UPnP

---

Minimum effect of function updates

- Even if a bundle is updated (or started),
  - No need for whole system (Java VM) reboot
  - Other bundles having no relation with the bundle will NOT be disturbed (run continuously).

1. Update Bundle(A)
2. stops A
3. install JAR again
4. starts A

- Without disruption
- Without rebooting
Comparison: Java Applications

<table>
<thead>
<tr>
<th>Java Applications</th>
<th>OSGi Bundles</th>
</tr>
</thead>
<tbody>
<tr>
<td>To run $N$ applications</td>
<td>$N$ bundles run on $1$ JVM (on the same process)</td>
</tr>
<tr>
<td>Start (or update) Appl.</td>
<td>(only 1 bundle load)</td>
</tr>
<tr>
<td>$N$ applications need to load same java package (e.g. library)</td>
<td>Package Sharing between bundles. (only 1 bundle load)</td>
</tr>
<tr>
<td>To get bundle cooperation</td>
<td>Service invocation through service registry of FW.</td>
</tr>
</tbody>
</table>

- Interprocess communication is required.

- Reduce memory consumption.
- Easy development.

Features of OSGi

- OSGi defines spec of
  - OSGi Framework (FW)
    - Bundle Lifecycle Management
    - Package Sharing
    - Service Registry
  - Standard services
- Policy Free and Open Remote Management
- Separation of Spec and Implementations
Policy Free and Open Remote Management

- OSGi Spec does NOT restrict **management policies** at all.
  - Gives platform operators considerable freedom to define their own policies.
  - Including **how to manage remotely**.

Separation of Spec. and Implementations

- Spec. defines only *functional requirements* and their *Java Interfaces* of
  - FW
  - Standard services.
- Spec is **fully open** to not only OSGi Alliance members but also others.
  - Anybody can implement FW and services.
  - Multiple vendors can implement for different optimization.
OSGi Framework: Commercial and Open Sources

- **Commercial products**
  - IBM, SUN Microsystems, ProSyst Software, Samsung, Atinav Inc., Connected Systems, Gatespace Telematics AB, Espial, Siemens VOD Automotive
  
  Many other companies are developing FW implementations for their own business. (not sold to others).

- **Open sources**
  - **Eclipse** ([www.eclipse.org](http://www.eclipse.org))
    - Current most commonly used Java IDE (Integrated Development Environment)
      - “Plug-in” architecture of Development modules
  - **OSCAR** ([oscar-osgi.sourceforge.net](http://oscar-osgi.sourceforge.net)) (*)
  - **Knopflerfish** (Gatespace, [www.knopflerfish.org](http://www.knopflerfish.org)) (*)
  - **Opensugar/JEFFREE** ([jeffree.objectweb.org](http://jeffree.objectweb.org)) (*)

- Many other companies are developing FW implementations for their own business. (not sold to others).

(*) has tutorial pages showing how to write bundles

---

**OSGi and related standards**

- **BROADBAND NETWORK**
  - FTTH
  - DSL
  - Cable
  - Wireless
  - Satellite

- **SERVICE DELIVERY**
  - OSGi

- **LOCAL NETWORK**
  - PLC
  - LonWorks
  - 802.11
  - HomeRF
  - JINI
  - UPnP
  - HomePNA
  - Bluetooth
  - Zigbee
  - HAVI
  - Echonet

---

Sep. 28, 2005 APNOMS2005 Tutorial

NTT
 Benefit of adopting OSGi

- Benefits of adopting OSGi as platform
  - For application developers, OSGi service platform providers, device manufactures
- in terms of
  - Reduced Development Cost
    - Open standardized API
    - Reuse software resources in multiple developments
  - Easy Customization
    - Service delivery by bundle installation
  - Easy deployment
    - OSGi Spec defines runtime environments of bundles.
  - Policy free and Open remote management
    - OSGi service platform operators can use OSGi for multiple purposes.

Outline

I. OSGi Alliance and its Background
II. Features of OSGi
III. Use Cases of OSGi
IV. Technical Details of OSGi FW
V. Standard Services in Spec. Release 3.0
VI. Management Issues for providers
VII. Summary
Business areas of OSGi

- Home Automation
- Industrial Automation
  - Factory Automation
  - Building Automation
- Remote maintenance of Home & Office equipment.
- Healthcare
- Support service for elderly and handicapped.
- Energy management, Ecology
- Telematics (Vehicle)
- Ubiquitous Applications
- Mobile phones
- …

Movements in Home Application Area(1)

- Use cases of commercial service is increasing
  - In Korea & China, condominium buildings
    - e.g. Mokdong Chereville (Korea)
      - Managing consumption of gas, electricity and water.
      - Air conditioner control.
      - Remote monitoring using intrusion sensors, fire alarms, and Web (security) camera.
      - Samsung plays primary role.
Movements in Home Application Area(2)

- “Shell Home Genie”: Consumer service for Shell and Motorola
  - Starter Kit is being sold.
    - OSGi based residential GW: produced by Motorola.
    - Cameras, door sensors, thermostats and power switches.
    - Multimedia services are the next stage.

  http://www.shellhomegenie.com/

iPronto Series of Philips

- Philips produces high class AV equipment mainly in USA and Europe.
- iPronto Series are “multiple functional remote controllers
  - http://www.pronto.philips.com/
  - OSGi FW on Remote Controller like tablet PC.
- By installation and activation of OSGi bundles
  - Control products newly bought.
    - Including new GUI.
  - Enable customization by Users
Movements of Telecom Carriers

- Telefonica
  - “Hogar.es” project
  - provides connected homes with a large variety of services in residential market.
    - remote management of electronic devices, security, home health care system with remote medical staff, and so on.
  - Pilot experiments.

- France Telecom
  - Has home portals mainly for multimedia applications: with Thomason Multimedia (Cocooning project).

- Deutsche Telekom
  - TeleHome project: intelligent home.
  - inHaus project: with Frauenhofer Lab and other many companies
    - built model houses.

- TeliaSonera
  - Delivering services to homes in a residential section of Stockholm
  - Locates GW functions not in each home but the center and controls home equipment through IPv6.

Movements of Industry Automation and Energy Control

- Mitsubishi Electric Group
  - Using their own implementation of OSGi FW “TSUBASA”
  - Factory automation of electric power plants, building automation,...

- EDF (Electric de France)
  - M@jordom project:
    - Show the electricity consumption with tariff and
    - Encourage lower electricity consumption at peak period
    - Home automation for efficient electricity consumption
    - Mainly from the viewpoint of ecology

- Energy control in many fields
  - Buildings, convenience stores, stations, and so on.

- Remote metering and management of electric and gas equipment
Movements in Telematics (Automotive)

- AMI-C adopted OSGi service platform (April 2003).
  - AMI-C (Automotive Multimedia Interface Collaboration): standardized APIs for mobile information and entertainment systems
    - Members: BMW, GM, Ford, Toyota, and other major car manufacturers.
- BMW 5 series (sold from 2004 1Q) adopts OSGi service platform
  - Currently used for telematics (car navigation system, telecommunication function)
    - In future, for automotive control too.
  - GSM/GPRS for Telecommunication
    - Future plan includes ad hoc communications using Wireless LAN.

Movements in Mobile phones

- Mobile Expert Group was established in December 2003
  - Objective: adopting OSGi as application platform of next generation smart phones.
- New additional specifications for mobile are being developed
  - Rely on NO existing specific specifications but possible to utilize them.
    - e.g. OMA (Open Mobile Alliance)
    - Will be main addition of OSGi Release 4.0.
- Have liaison with JCP (JSR-232)
  - JCP (Java Community Process): participative process to develop and revise the Java technology specifications.
  - JSR: Java Specification Request.
Eclipse 3.0 adopts OSGi

- The most famous open source IDE (Integrated Development Environment)
  - Mainly for Java
  - Originated inside IBM.
- Small extensive runtime core and
- Functionalities, including GUI, the workbench, the compilers, consist of plug-ins.
- Eclipse 3.0 adopts OSGi service platform as a base of runtime core (June, 2004)
  - Since version 3.0, adding or updating a plug-in in Eclipse requires NO rebooting.

Outline

I. OSGi Alliance and its Background
II. Features of OSGi
III. Use Cases of OSGi
IV. Technical Details of OSGi FW
V. Standard Services in Spec. Release 3.0
VI. Management Issues for providers
VII. Summary
OSGi Fundamental Technologies

- Bundle Lifecycle Management
- Package Management
- Service Registry
- Security
- Event notifications
- Persistently stored information
- Native codes wrapped in a bundle

Bundle

- JAR (Java Archive) format file containing
  - Java classes to execute.
  - Resources to be used by the executing Java classes.
  - Manifest file with headers specified by OSGi Spec
    - With required info in order for FW to install and activate the bundle correctly.
      - `BundleActivator` implementing class (described later)
      - `Dependencies of Java packages` (described later)
      - `Copyright, Name, Description, Vendor, Version, ...`

**Bundle1**

```
META-INF/MANIFEST.MF
apnoms/tutorial.osgi/Intro.class
apnoms/tutorial.osgi/TestActivator.class
apnoms/tutorial/qos/Detail.class:
res/image.gif
```

- Manifest-Version: 1.0
- Bundle-Activator: apnoms.tutorial.osgi.TestActivator
- Export-Package: apnoms.tutorial.qos
- Bundle-Description: APNOMS 2005 Tutorial
- Bundle-Copyright: NTT Copyright
- Bundle-Vendor: NTT
- Bundle-Version: 1.2.3
- Bundle-ContactAddress: yamasaki.ikuo@lab.ntt.co.jp

- FW is also a Bundle “system bundle”.
Life Cycle Management of Bundles

- MA can Install Bundles, and Start, Stop, Uninstall, and Update, dynamically.

```
INSTALLED

UNINSTALLED

RESOLVED

STARTING

STOPPING

ACTIVE
```

Bundle State Diagram

Bundle Install

- FW installs a bundle when MA calls
  `org.osgi.framework.BundleContext.installBundle(String bundleLocation)`
  - FW retrieves JAR file from the specified `bundleLocation` and installs it as a bundle.

- Bundle Identifier on a FW
  - `bundleLocation`: unique identifier of bundle on a FW.
    - Usually URI of the bundle's JAR file.
      - E.g. file://hoge/hoge.jar or http://osgi.ntt.co.jp/hoge/hoge.jar
  - `Bundle ID`: Integer value allocated by FW to each bundle at its installation incrementally.
    - 1,2,3,...Integer.MAX_VALUE.
    - NO change even if updating the bundle or rebooting FW.
    - Bundle ID of System Bundle (FW) = 0.
BundleActivator interface

- Bundle usually has a class which implements org.osgi.framework.BundleActivator (Java interface)
  
  - Start(BundleContext )
    - Initializing operations should be implemented.
      - e.g. Register service to the service registry.
      - Create threads required for later operations.
  
  - Stop(BundleContext )
    - Cleanup operations should be implemented.
      - e.g. Kill all threads created by the bundle.
      - (FW automatically un-registers services that the bundle registered)

Manifest headers of a bundle

- At installation of a bundle, FW reads required info in manifest headers to install and activate the bundle correctly.

  Bundle1
  META-INF/MANIFEST.MF
  apnoms/tutorial.osgi/Intro.class
  apnoms/tutorial.osgi/TestActivator.class
  apnoms/tutorial/qos/Detail.class
  res/image.gif

  Manifest-Version: 1.0
  Bundle-Activator: apnoms.tutorial.osgi.TestActivator
  Export-Package: apnoms.tutorial.qos
  Bundle-Description: APNOMS 2005 Tutorial
  Bundle-Copyright: NTT Copyright
  Bundle-Vendor: NTT
  Bundle-Version: 1.2.3
  Bundle-ContactAddress: yamasaki.ikuo@lab.ntt.co.jp

  FW can identify which class implements org.osgi.framework.BundleActivator.
Bundle install & start

(*)Many operations are omitted for easy understanding.

**MA bundle**

- BundleContext.installBundle(location)
- Bundle.start()

**FW**

- Download JAR from location and install as a bundle.
- Create Bundle object and BundleContext object for the installed bundle.
- Return the Bundle object to requesting MA bundle.
- Read Manifest Header of the bundle and identify BundleActivator class
- Create BundleActivator instance for the bundle
- BundleActivator.start(BundleContext)

**BundleActivator**

- Implementation of start(BundleContext) is called

---

Bundle stop & uninstall

**MA bundle**

- Bundle.stop();
- Bundle.uninstall();

**FW**

- BundleActivator.stop(BundleContext)
- BundleActivator.uninstall(BundleContext)
- Cleanup
  - Bundle object and BundleContext object and BundleActivator are reset to null.

**BundleActivator**

- Implementation of stop(BundleContext) is called

---
Java Package

- A group of java classes.
- To manage many classes, classes are grouped into hierarchical directories.
  - The same (simple) class name can be used in different packages.
  - A class is identified by Fully Qualified Name.
    - Package name + "." + simple class name.

```
Package Name    Simple Class Name
tutorial.osgi.Intro

Fully Qualified Class Name
tutorial.qos.Intro
```

Identified as Different classes

General Java rule: package name start with a lower case letter,
and simple class name starts with an UPPer case Letter.

Class Loading / Package management

- Each Bundle has separate classloader
  - Each bundle classloader has independent class name-space.
  - Avoid class name conflict between bundles;
    - Bundle developer can design its packages and class independently. (Different bundles can load its own class with same name).

```
Bundle1
jp.co.ntt.osgi.Test(class)

Bundle2
jp.co.ntt.osgi.Test(class)
```

Each will be loaded as different class.

OSGi FW

JVM
**Package Sharing** between bundles.

- Bundles can import a package exported by a other bundle.
  - Must have "Export-Package( or -Import)" manifest header.
  - Reduce consumption of system resources (memory & disc space).

**Merit of Package Sharing**

- Export Commonly used Java library
  - Bundle1
    - Package `jp.co.ntt.osgi.util`
    - Export package

- Import `jp.co.ntt.osgi.util`
  - Bundle2
    - Package `jp.co.ntt.osgi.util`
    - Import package
  - Bundle3
    - Package `jp.co.ntt.osgi.util`
    - Import package

**Class Loading / Package management**

- In "META-INF/Manifest" Header
  - Bundle1
    - Export-Package:`p.p1`
  - Bundle2
    - Import-Package:`p.p1`
Service

- Defined semantically by its service interface.
  - With N (1 < N) methods (without implementation).

- Implemented as a service object.

- Unit of functionality provided by a bundle and used by other bundles through the service registry.
  - FW manages the service registry.
Service Registry: Register service

1. Create a service object implementing specified service interface.
2. Create properties of it.
   - Pairs of (key, values)
   - Keys: "service.pid" / "service.description" / "service.vender" / anything is defined by developers.
3. Register the service object as the service name with the properties into the service registry.
   - BundleContext.registerService(serviceName, ServiceObj, prop);
   - Multiple service objects can be registered as the same service name with Different properties.

Service Registry: Use service

1. Search service objects in service registry and choose one.
   - Create LDAP filters to search based on service name and properties.
2. Get service object from service registry.
3. Call its methods defined by its service interface.
Separation of spec and implementations

For developing an original service,

- Define service spec and open it to both developers of service registering bundles and using bundles
  - Java interface and its functional requirement.

Independent Development of bundles:

- developers on either side do NOT need to know anything about others’ implementation only service spec.
- Especially, developer of bundles that use the service can develop it with NO knowledge of service object implementation.
- Even if source code of bundle registering a service is rewritten (e.g. for bug fix), no need to rewrite bundles that use it.

Service Registry and Package sharing

Case1: A bundle exporting service interface and a bundle registering service object is same.

In “META-INF/Manifest” Header
Export-Package: p
In “META-INF/Manifest” Header
Import-Package: p

Separate Development of bundles:

developers on either side do NOT need to know anything about others’ implementation only service spec.
- Especially, developer of bundles that use the service can develop it with NO knowledge of service object implementation.
- Even if source code of bundle registering a service is rewritten (e.g. for bug fix), no need to rewrite bundles that use it.
Service Registry and Package sharing

Case 1: A bundle exporting service interface and a bundle registering service object is same.

1. Create service object of "p.Service1" (Service1Obj)
2. Create properties of it (prop)
3. registerService("p.service1", Service1Obj, prop)
4. Create filter of properties to search (filters)
5. getService("p.Service1", Service1Obj, filters) (*)
6. Invocate methods of "p.Service1".

(*) For understanding, this Image shows incorrect method name - signature.

Case 2: A bundle exporting service interface and a bundle registering service object is different.

No Problem !!
The same service interface must be loaded by registering bundle and using bundle.

Service Registry and Package sharing

Export-Package: p
Import-Package: p
Import-Package: p

Bundle1
Package p

Bundle2
Package p
Service1Obj

Bundle3
Package p
Service1Obj

OSGi FW

Sep. 28, 2005 APNOMS2005 Tutorial
OSGi Fundamental Technologies

- Bundle Lifecycle Management
- Package Management
- Service Registry
- Security
- Event notifications
- Persistently stored information
- Native codes wrapped in a bundle

Security: PermissionAdmin service

- General Java2 Runtime Permission Check:
  - Enable control of permissions granted to applications.
  - Applications are checked as to whether the required permissions have been granted or not at runtime.
  - E.g. In order to
    - Restrict area of local file systems to read, write and execute.
    - Restrict Socket usage with host, port number, and action (connect, listen, …)
- In OSGi, PermissionAdmin enables MA to control permissions granted to each bundle
  - one of standard services defined by the Spec.
    - Registered by only FW.
    - Since Release 2.0
PermissionAdmin service

- Three OSGi original permissions are defined by Spec.
  - **AdminPermission**: is required for critical operations
    - e.g. lifecycle management of bundles, set Permissions, and so on.
  - **PackagePermission**: is required to import or export specified package.
    - e.g. (PackagePermission “p” “IMPORT”) is required for a bundle to import package "p".
  - **ServicePermission**: is required to register or get a specified service.
    - e.g. (ServicePermission “p.Service1” “GET”) is required for a bundle to get service object “p.Service1”.

PermissionAdmin service

- Enables MA to protect against
  - Unauthorized bundles -- install / start/ stop / uninstall / update bundles.
    - By AdminPermission
  - Unauthorized bundles -- export packages, which may include inappropriate or malicious code.
    - By PackagePermission
  - Unauthorized bundles -- register service objects, which might be inappropriate or malicious implementation, into Service Registry.
    - By ServicePermission
  - e.g. operation to set Permissions through PermissionAdmin requires all of the below
    - (AdminPermission);
    - (PackagePermission, “org.osgi.service.permissionadmin”, “IMPORT”);
    - (ServicePermission,
      “org.osgi.service.permissionadmin.PermissionAdmin”, “GET”);
Event Notifications

- FW provides dynamic environments of
  - Bundles, services, packages, ...
- FW publishes important Events when states change.
  - Listeners can catch these Events.
    - **FrameworkEvent** and -Listener
      - STARTED / ERROR / PACKAGE_REFRESHED / STARTLEVEL_CHANGED.
    - **BundleEvent** and (Synchronous) -Listener
      - INSTALLED / STARTED / STOPPED / UNINSTALLED / UPDATED.
    - **ServiceEvent** and -Listener
      - REGISTERED / MODIFIED / UNREGISTERED

Bundle Dependency : Package

- When sharing package,
  1. Bundle $A$ exporting package “$p$”,
  2. Bundle $B$ importing package “$p$” : successful resolution.
- Next, bundle $A$ is uninstalled or updated.
  - Old exported package “$p$” can still be used by bundle $B$.
- Mainly two kinds of policies are possible.
  - [Policy 1 ; Eager Update]: stop bundle $B$ and try to resolve and start() (in case of update) again.
    - Old exported packages never used later.
  - [Policy 2 ; Lazy Update]: until FW is rebooted, let bundle $B$ use old exported packages.
Package Admin service

- **PackageAdmin** is one of the standard services defined by the Spec (Since Release 2.0).
  - registered by only FW.
- MA can control the policies at package dependency changes.
  - **[Eager update policy]**
    - Register *(Synchronous)*BundleListener
      - When BundleEvent.UNINSTALLED or UPDATED is hooked, refresh exported packages by the uninstalled or updated bundle through PackageAdmin
        - Stops bundles that import the packages and tries to resolve and start (if the bundle state used to be ACTIVE).
  - **[Lazy update policy]**
    - Do nothing.
  - Other policies can be implemented by MA.

Bundle Dependency: Service

- Dependency between bundles registering and using the same service is important.
  - If a bundle using a service tries to get service object from the service registry before it is registered (by other bundle), it fails to get service (results in error).
- A bundle to get a service object and use it MUST be implemented to track the status of the intended service and do appropriate operations.
  - E.g.: If the intended service is registered, then a bundle gets the service object. If unregistered, releases it.
Service Tracker

- [Measure 1] use **ServiceListener**.
  - Enables a bundle to hook **ServiceEvent**.
    - REGISTERED / UNREGISTERED / MODIFIED
  - Developers should implement their own filter codes that pass only intended **ServiceEvent**, because **ServiceListener** doesn’t have filter functions by **service properties**.
  - Moreover this programming model is error-prone.

- [Measure 2] use **ServiceTracker**.
  - Since 2.0, **ServiceTracker** provides utilities to track the intended **service**.
    - Developer can set filters, in same way as service registry’s filter: LDAP filter.
    - Much easier to implement tracking **services**.
    - STRONGLY RECOMMENDED

Service Registry used for Event Handling

[ Programming TIPS]

- Bundle developers do NOT need their own event handling mechanisms, **Event Publishers and Listeners**.
- The Whiteboard pattern
  - Easy development of Event Handling applications.

Can control timing and contents (arguments of a method) to pass the listener.
Persistently Stored Information

- **FW should hold persistent information.**
  - When booting, FW tries to reconstruct the situation as at the last FW shutdown.
    - Bundle Information.
      - Bundle Jars are stored in persistent storage by FW. When booting, FW installs them from the local storage (and starts)
        - No Bundle IDs or bundle locations change.
    - PermissionAdmin Information
    - ConfigurationAdmin Information (described later)
    - UserAdmin Information (described later)
    - Others...

Native codes Wrapped in a Bundle

- Multiple native codes each of which depends on an OS can be wrapped in a bundle.
- “Bundle-NativeCode” Manifest Header.
  - FW reads it and decides which native code to execute.

```java
Bundle-NativeCode: /lib/http.DLL ;
/lib/zlib.dll ;
osname = Windows98 ;
osname = WindowsNT ;
processor = x86 ;
language = en ,
/lib/solaris/libhttp.so ;
osname = Solaris ;
osname = SunOS ;
processor = sparc ,
/lib/linux/libhttp.so ;
osname = Linux ;
processor = mips
```
Outline

I. OSGi Alliance and its Background
II. Features of OSGi
III. Use Cases of OSGi
IV. Technical Details of OSGi FW
V. Standard Services in Spec. Release 3.0
VI. Management Issues for providers
VII. Summary

OSGi Spec Release 3

- Device Manager
- Http Service
- Log Service
- Preferences Service
- Configuration Admin Service
- Service Tracker
- User Admin Service
- Wire Admin Service
- XML Parser Service
- Measurement and State
- Position
- Jini Driver Service
- UPnP Device Service
- Package Admin Service
- Permission Admin Service
- Start Level Service
- URL Handler Service
- Framework

Already explained

Services provided by FW

Sep. 28, 2005 APNOMS2005 Tutorial 75
Start Level service

- When FW boots, FW installs bundles stored in local persistent storage.
- **The order of starting bundles**
  - Default:
    - Increasing order of bundle ID
    - = The order of initial bundle installation.
  - Using **Start Level** service:
    - MA can control the order of starting bundles at FW boot freely.
    - E.g., For **safe mode** after some trouble.
- **Since Release 3.0**
- Registered by only FW.

Configuration Admin service

- Most bundles and services need to remember and update *configuration* information.
  - E.g., IP address and port number of server
- **Configuration Admin** provides unified mechanisms to
  - Set and update *configuration* information, which is stored persistently.
  - Read the stored information.
- **Since Release 2.0**

```
[Diagram showing the interaction between Configuring Bundle, Configuration Admin, and Configuration target Bundle]
```

Sep. 28, 2005
APNOMS2005 Tutorial
### Services defined since OSGi Release 1

- **Device Access**
  - When plug and play device is attached, Device Access provides mechanisms that allow a bundle implementing its driver of the device to be automatically installed and started.

- **Http Service**
  - HTTP server (servlet runner) can be provided by this service on the service platform. Bundles can register servlets on it and HTTP clients can access its service via HTTP.

- **Log Service**
  - Enables any bundle to log information, warning, debug, or error.
  - Bundles that can read logged data can be controlled by MA through PermissionAdmin.

### Other Services since Release 2

- **User Admin service**
  - Provides a database of user information for authentication and authorization.
    - The authentication method is not restricted by the spec.
    - Grouping can be used for authorization.
    - Persistently recorded.

- **Preference service**:
  - Provides hierarchical tree database of properties and enables bundles to create and obtain a tree, to store and retrieve them in a tree.
    - E.g. for a game application,
      - Preference setting of each user
      - Highest scores among users in a system
    - Persistently recorded.
Other Services since Release 3

- **URL Handler service**
  - Enables bundles to provide new schemes or content handlers to URL class dynamically.
- **XML Parser service:**
  - Enables bundles to parse XML.
- **Metatype**
  - Defines interfaces that allows bundle developer to describe attribute types in Meta-data.
- **IO Connector Service**
  - Enables bundles to add and change new protocol schemes for the J2ME javax.microedition.io package.
- **Wire Admin service**
  - Provides mechanism to pass data between bundles, Producer and Consumer.
- **Jini Driver service**
- **Upnp Device Service**
- **Measurement and State**
- **Position**
  - Maps each protocol to OSGi services.
  - Utilities for vehicles

Summary

I. OSGi Alliance and its Background
II. Features of OSGi
III. Use Cases of OSGi
IV. Technical Details of OSGi FW
V. Standard Services in Spec. Release 3.0
VI. **Management Issues for providers**
VII. Summary
Management Issues

- OSGi doesn’t define remote management functions,
- There are a lot of management issues that platform operators and service providers have to deal with, in order to provide services to end users!!

Not OSGi services,
But ones that customers notice

Service Aggregation Platform (1)

- Proposed by NTT Lab.

SAP (Service Aggregation Platform)
- Operation center and Home gateway are shared by multiple service providers

For each user, only 1 access point and 1 home gateway
Service Aggregation Platform (2)

- Merits for every player.
  - Focus on their service development
  - Timely service provision
  - One stop access by users
  - Added to access line subscription fee, Platform business income.
  - Reduce space & cost

- Focus on their service development
- Operation Center (Portal site)
- OSGi FW
- Home Gateway

- Web based operations
- Low initial and running cost for individual service providers.
- Cost of operation center and HGWs

Service Aggregation Platform (3)

- Management issues
  - Managing service subscription between service providers and users
  - Managing distributions of bundles according to users’ subscription.
    - Including conflicts between multiple service providers’ services.
  - Monitoring status of HGWs
    - Not only OSGi elements but also others on different layers.
  - Separation of database for individual service providers.
  - Customization for individual service providers.
  - and so on.
Outline

I. OSGi Alliance and its Background
II. Features of OSGi
III. Use Cases of OSGi
IV. Technical Details of OSGi FW
V. Standard Services in Spec. Release 3.0
VI. Management Issues for providers
VII. Summary

Summary

- The OSGi spec. provides a comprehensive operating environment for Java applications.
  - Collaborative model
    - Java package sharing
    - Service Registry
  - Open specifications
    - Separation of specifications from implementations
    - Third parties can define an original service and implement it easily.
  - Policy free and Open remote management
- OSGi is beneficial to all players!!
  - Application developers, OSGi service platform providers, device manufactures, and customers.
If you get interested in OSGi

  - Get the whitepaper and the Spec of OSGi. It's free!!

- If you have experience in Java programming,
  - read tutorial web pages to code bundles
    - Knopflerfish: [http://www.knopflerfish.org/programming.html](http://www.knopflerfish.org/programming.html)
  - Also the following whitepaper is worth reading.

Thank you for your attention !!