OSGi Architecture and IoT

Tim Ward - IoT EG chair
tim.ward@paremus.com
OSGi is built around modules called **bundles**
A bundle runs inside an OSGi runtime called a **framework**
Bundles are JAR files with metadata
The metadata lives in MANIFEST.MF

The metadata defines:

The bundle identity

The bundle contracts

The bundle requirements

Bundle

MANIFEST.MF
Manifest-Version: 1.0

Bundle-SymbolicName: org.example.bundle
Bundle-Version: 1.2.7.RELEASE

Export-Package: org.example.foo;version=1.1
Provide-Capability: osgi.extender;
    osgi.extender=org.example.foo;version=1.2

Import-Package: org.apache.log4j;version=[1.2,2)
Require-Capability: osgi.ee;
    filter:="(&(osgi.ee=JavaSE)(version=1.8))"
So why is this good for IoT?

Modularity is great for heterogenous systems
  Need to add a device using a new standard?

Modularity makes it easy to add features and fix bugs
  Need to add a new metrics reporting transport?
  Want to update a driver without disrupting everyone else?

Modularity makes it easy to control what you run
  Are we using this version of library X that has a security vulnerability?
Truly decoupled modules must share objects by interface
How do you get an object when you can only see its interface?

The OSGi Service Registry enables sharing these objects
OSGi Services

Using services is a powerful way for bundles to collaborate

Services advertise their public interfaces
  This provides a type-safe way to identify what a service does

Services can be registered and unregistered dynamically
  Clients can be notified of these changes

Services have properties for information and filtering
  Useful for identifying sub-categories or specific services
So why is this good for IoT?

Using services makes it easier to replace implementations
Switching from oneM2M to Web of Things?

Services give a common discovery pattern
However something is connected you communicate in the same way
Using ten things is no different from using one

Services are dynamic, just like things are
Services register when new things are detected
Services unregister when things run out of battery/go out of range
So why is this good for IoT?

OSGi implementations can be very lightweight
   Many frameworks are only a few hundred kilobytes

OSGi is an excellent technology to deploy at the edge
   Remotely manageable and updatable
   Good for connecting to different things locally

But what about elsewhere?
   Does OSGi only make sense at the edge…
Truly decoupled modules must share objects by interface

If you’re truly decoupled does it matter who the provider is?
OSGi services can come from anywhere!
Any service can be transparently remoted *
Service properties are made available remotely too

* Make sure your service is actually suitable for remoting!
Exposing a remote service is not transparent

Services opt-in using the `service.exported.interfaces` property

Imported services are marked with a `service.imported = true`
So why is this good for IoT?

Remote Services provide transparent remoting
Function can be provided centrally or pushed out to the edge

Transports and discovery technologies are pluggable
Different implementations possible for different layers of the system

Native support for Asynchronous remoting
A rich API possible using Promises

OSGi provides a simple path from edge to fog to cloud
IoT specific activities in the OSGi Alliance

The OSGi Alliance has several Expert Groups

The IoT Expert Group focuses on IoT relevant use cases

Current RFCs include

- The oneM2M Interworking Service
- The oneM2M Service Layer API
- MQTT Protocol adapter

OSGi Requirements Documents are publicly visible

https://github.com/osgi/design
Interacting with oneM2M

Integration is about being part of oneM2M networks
How does my OSGi bundle register as a oneM2M device?
How does it receive notifications when a Content Instance changes?

The Service Layer is for interacting with oneM2M devices
What devices are available?
What actions does that device have?
Change that device from “OFF” to “ON”

These specifications are deliberately separated
It’s rare to be a device and a management agent at the same time!
Working with MQTT

MQTT is a lightweight asynchronous messaging standard
Many Smart devices communicate using MQTT

OSGi supports asynchronous actions and event streams

This RFC investigates ways to bridge these areas
Summary

OSGi is a powerful technology used across many sectors

Modularity is a powerful advantage at the edge

The OSGi service model is a great abstraction for IoT

OSGi solutions are in place right now

And it’s time for you to hear about some of them!