

OSGi and SOA

Understanding how OSGi helps SOA

January 2009

Paul Fremantle
Co-Founder & CTO

Paul Fremantle

Co-Founded WSO2 in 2005

- Aiming to be the Open Source SOA platform of choice

Apache Member

- VP, Apache Synapse
- PMC: HTTPComponents, Incubator, Qpid, Tuscany

Co-chair WS-RX Technical Committee, OASIS

Infoworld CTO 25

OSGi History

OSGi initially founded in 1999 to make it easier to deploy Java applications in embedded devices

In 2003 Eclipse chose OSGi as the plugin architecture

OSGi R4 released in 2005

May 2007 - OSGi R4.1 released (JSR 291)

OSGi and Middleware

- Originally designed as a modularity model for embedded and mobile devices
- Increasingly used as a framework by Enterprise Middleware
 - Spring dm Server
 - IBM WebSphere
 - JBoss AS etc
- “SOA for the infrastructure”
 - using a Service model within a JVM
- James Governor from Redmonk calls OSGi the
 - “Stackless Stack”

What does OSGi give me?

- **Modularity:**
 - Each JAR is a **Bundle**
 - Each bundle has its own classloader
 - Bundles can share or hide packages
 - Import a package, or require another bundle (import all packages)
 - Bundles have versions
- **Services**
 - Each bundle can provide *services* to other bundles
 - Services are simply Java objects that implement a given interface
 - There is a Service Registry where you can find a service implementation
- **Dynamic loading and lifecycle**
 - Bundles may be stopped, started, uninstalled and updated without stopping the framework

OSGi Bundle MANIFEST.MF

Manifest-Version: 1.0

Bnd-LastModified: 1229014618171

Export-Package: org.apache.http;uses:="org.apache.http.params,org.apache.http.util,org.apache.http.protocol",.....

Bundle-Version: 4.0.0.beta1

Bundle-Name: org.wso2.carbon.httpcore

Bundle-Description: org.wso2.carbon.httpcore. This bundle will export packages from httpcore.jar

Build-Jdk: 1.5.0_14

Bundle-ManifestVersion: 2

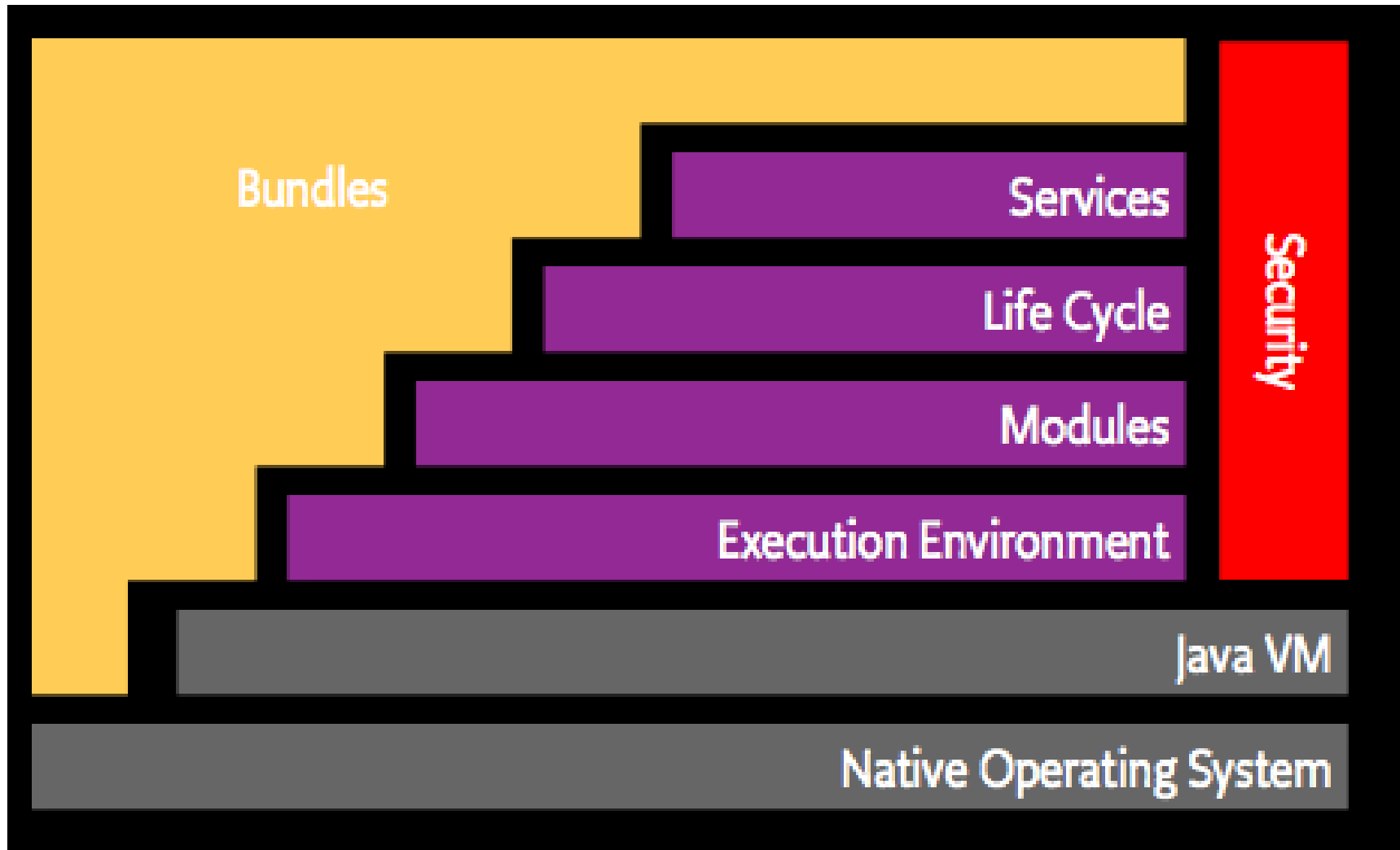
Bundle-Vendor: WSO2 Inc

Bundle-SymbolicName: org.wso2.carbon.httpcore

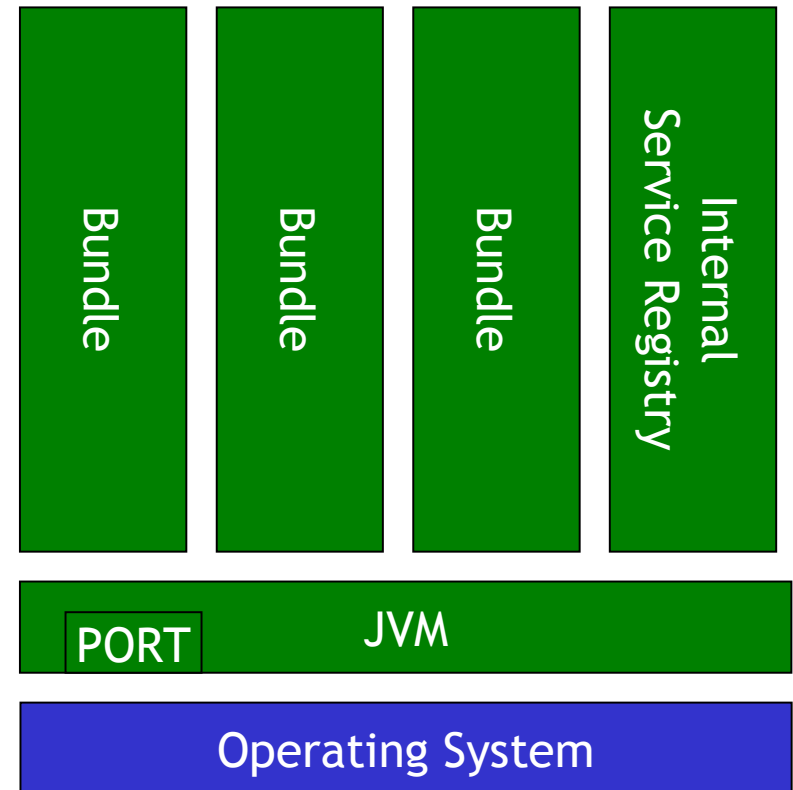
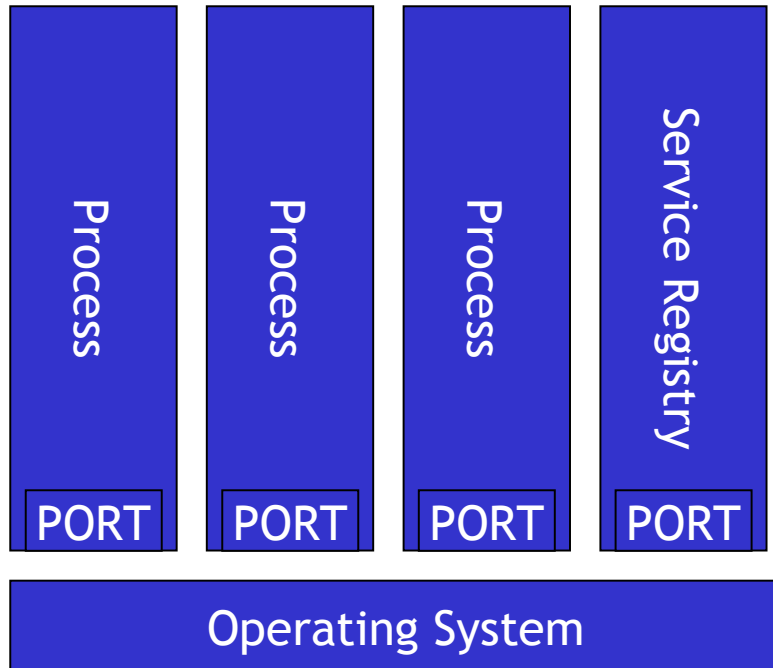
Tool: Bnd-0.0.238

WSO2-Bundle-StartLevel: 20

OSGi layering



Integration at the OS/network or the JVM?



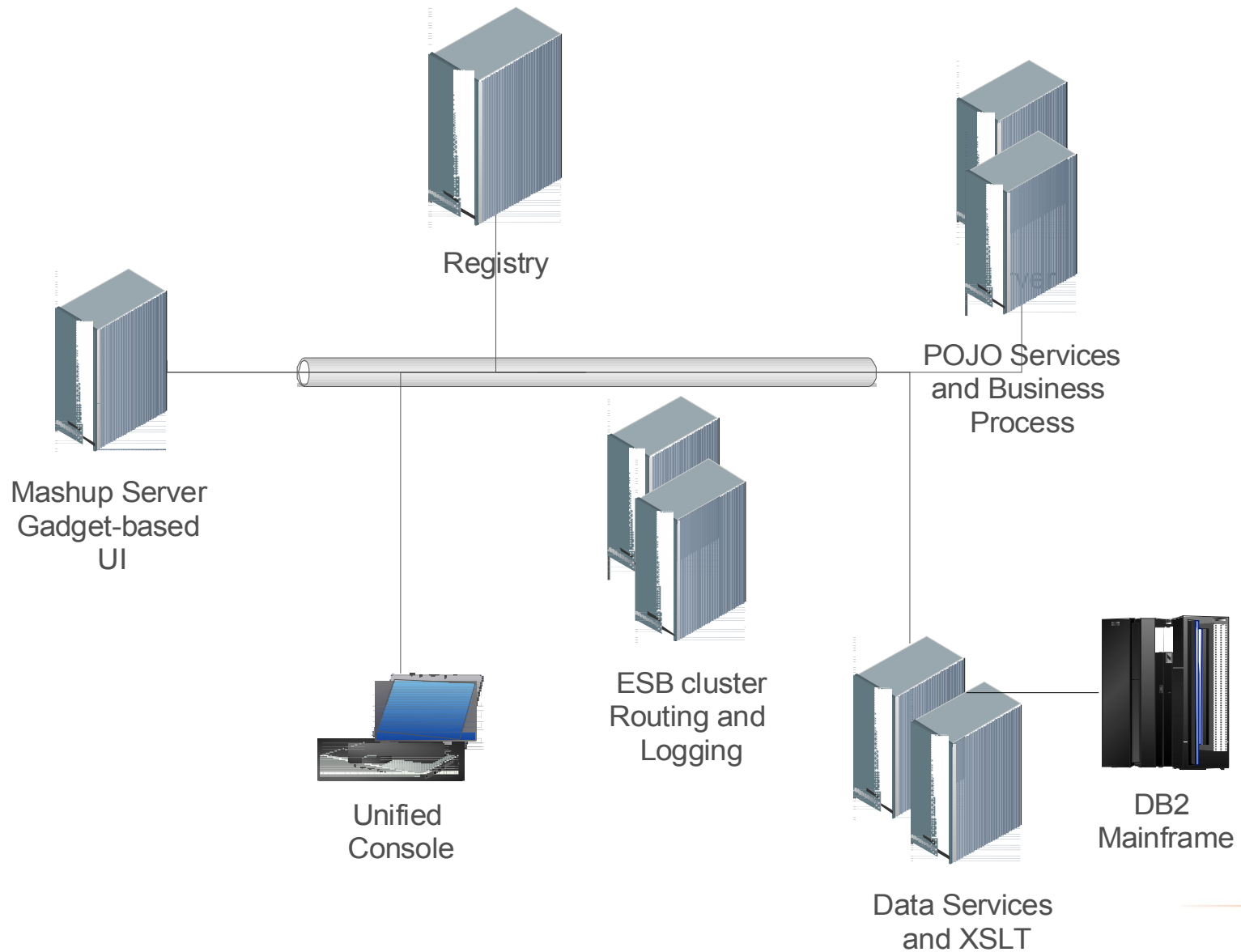
SOA Anti-patterns

- Traditional / proprietary SOA platforms are:
 - Overly complex
 - Heavyweight
 - Not internally consistent
 - Hard to learn
- The result is:
 - Customers centralize SOA on a single hub based ESB
 - Departments rely on the “SOA team” to solve problems
 - Replacing centralized monolithic applications with
 - Centralized monolithic ESB and SOA platform
- What should we be doing?
 - Allowing departments to host *just enough* SOA infrastructure
 - Making it easier to start simple and grow as needed
 - Making the SOA platform integrate simply and naturally

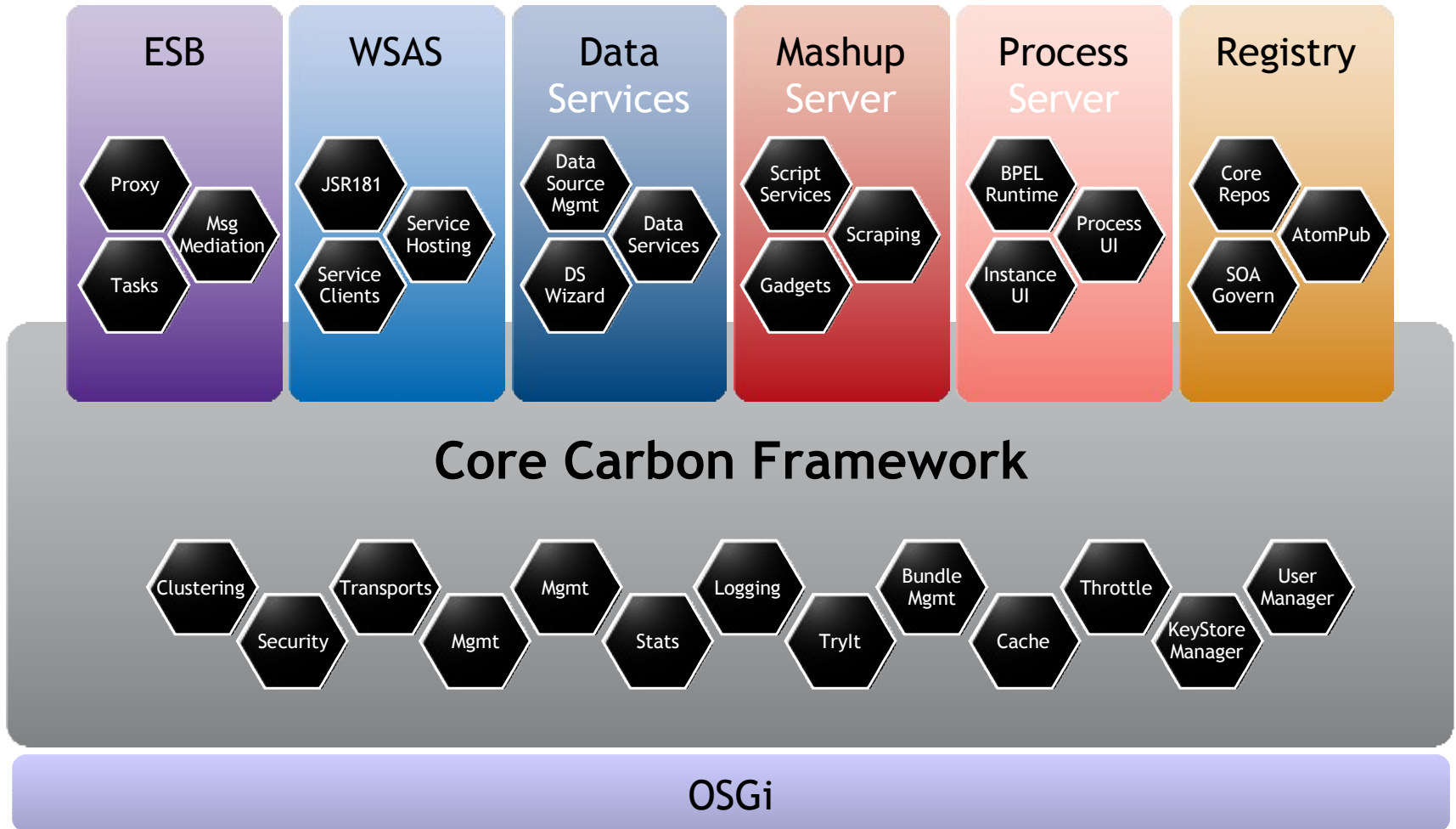
How does OSGi help?

- Suppose each part of the SOA puzzle was available as OSGi components:
 - Mediation & ESB
 - POJO Service Hosting
 - BPEL Process flow
 - Data Services
 - Registry
- And these composed naturally with other components:
 - User and Security management
 - Test capabilities
 - Logging and Audit
 - Reliable Messaging
 - Monitoring
- As well as other components:
 - Apache ActiveMQ
 - OpenFire XMPP Server
 - etc

A distributed example



WSO2 Carbon 1.5



How Carbon uses OSGi

- OSGi is the underlying core modularization technology
 - Shipping with **Eclipse Equinox** by default
 - Can be supported on Spring dm Server, Felix and Knoplerfish if required
- Uses the core OSGi approach to support plugging in new function in a managed way
 - Versioning
 - Clean separation of concerns
- Any OSGi bundle can be deployed (e.g. ActiveMQ)
- New components can be deployed into an existing installation
- Used for both server runtime as well as for componentized management console
- Customers can write and deploy their own OSGi components

Carbon is more than just OSGi

- Analogies:
 - Eclipse uses OSGi but is much more - it defines how you plug Tool components into a Tooling Framework
 - Microsoft Office uses DLLs, but is much more - it defines how you can share a graph component across multiple document types
- Carbon uses OSGi, but it also defines how you build a consistent SOA platform
 - New service types plug into the console
 - Security, throttling, stats, TryIt, all work on any service component
- Not just using OSGi for componentizing a single product, but rather *entire* middleware platform: app server, ESB, process server, registry, governance, mashup and more
- Our “products” are now simply our choice about how you start with this platform
 - You can assemble your own “product” by combining components

Roadmap

- September 2008
 - Carbon 1.0 release (Data Services)
 - Internal WSO2 proof point
- Feb 6th 2009
 - Carbon 1.5 release (ESB, WSAS, BPS, Registry)
 - First full OSGi release
- Feb 28th 2009
 - Feature Packs available:
 - Mediation, BPEL, Java Service Hosting, Cluster Management
- Mid 2009
 - OSGi as a development and deployment model
 - Axis2 services packaged as bundles
 - Mediators packaged as bundles, etc
 - Dynamic support for reloading code and config

Lessons learnt about OSGi

Benefits:

- Better management of classloading
 - Imports and Exports
 - Proper Modularity
 - Versions of modules and dependencies
- Supports multiple versions of the same code
 - As long as there are no cycles
- Supports dynamic loading/reloading
 - Upgrade your code or a system in flight
 - Not yet used by Carbon

Problems:

- Existing code using classloaders is a big issue
 - Either change the code or the classloader
 - PAX tools can help
- No built in support for JSPs in the model
 - Plenty of well-documented workarounds on the Web
- Getting the “start level” of bundles right is hard
 - <http://ruwansblog.blogspot.com/2008/10/osgi-bundle-start-levels.html>
- Building and managing patches is a significant issue
 - <http://afkham.org/2008/10/how-to-create-patches-for-osgi-bundles.html>

Find out more about Carbon:

Upcoming Webinar:

WSO2 Synergies: the Carbon Story

Feb 17th 2009

<http://wso2.on.intercall.com>

Questions?