1. Introduction
    1.1. Project/Component Working Name:
        OSGi and GlassFish

    1.2. Name(s) and e-mail address of Document Author(s)/Supplier:
        Sahoo: Sahoo@Sun.COM
        Richard Hall: richard.s.hall@oracle.com

    1.3. Date of This Document:
        24 June 2010

2. Project Summary
    2.1. Project Description:
        In this document, we describe OSGi [1] related features supported in GlassFish application server version 3.1. This is a continuation of our effort which had begun during GlassFish V3 Prelude release and continued in V3. Please refer to our earlier proposals [1] and [2] for a background to the problem. You may find certain differences between earlier proposals and the current one.

    2.2. Risks and Assumptions:
        Same as before.

3. Problem Summary
    3.1. Problem Area:
        Today's enterprise Java applications have a growing need for sophisticated dependency management, better lifecycle management and a dynamic service platform. Hence applications would like to use OSGi to meet some of these requirements.

        There is a need for an embeddable GlassFish that has not only the same features as non-embedded GlassFish, but also the capability to take advantage of OSGi eco-system.

    3.2. Justification:
        We need to provide a platform that meets requirements of a new generation of enterprise applications.

        As we expose OSGi as a first-class user visible feature in GlassFish, we need to provide better administration support for the OSGi runtime.

4. Technical Description:
     4.1. Details:
        The features proposed in this document fall into following categories:
        a) Core OSGi runtime:
           GlassFish will continue to use and distribute Felix as the default OSGi runtime. Currently, GlassFish uses Felix 2.0.2. We shall upgrade to Felix 3.0.x. Felix 3.0 (yet to be released) has a new resolver which is not only faster, but also has many important bug fixes.

        b) Support for hybrid (OSGi enabled Java EE) applications:
           There are two parts to this, viz:
           b.1) Enterprise OSGi Specs:
                We plan to implement the following enterprise OSGi specs in GlassFish V3.1:
                OSGi/HTTP Service
                RFC 66: OSGi/Web container
                RFC 98: OSGi/JTA
                RFC 122: OSGi/JDBC
                RFC 142: OSGi/JNDI
                RFC 143: OSGi/JPA
                Starting with V3 release, we already have an implementation of

OSGi/HTTP Service implementation and RFC 66. In this release, we shall make our implementation compliant with the spec. We may look at the possibility of integrating a third-party OSGi blue-print implementation in GlassFish.

b.2) OSGi & Java EE integration (Experimental)
We shall explore use of other EE APIs like EJB, JPA, JAX-RS, CDI in hybrid applications.
OSGi/EJB integration:
Users can deploy OSGi bundles containining Enterprise Java Beans and the EJBs will be deployed to GlassFish EJB container. Users can configure their hybrid application manifest file to selectively export EJBs to the OSGi service registry. The manifest header will look like this:

Export-EJB: <names of EJBs>

User can use a special value "ALL" to export all EJBs. At this point of time, we can only export stateless and singleton EJBs to OSGi service registry because of mismatch in OSGi service and EJB life cycle models. Similarly, since we don't have a concept of distributed OSGi service registry, we will only expose business interfaces of EJBs in local OSGi service registry.

OSGi/JPA integration:
Enterprise OSGi specs do not cover EE style use of JPA in hybrid applications. We shall support this use case in GlassFish. We will also provide support for standalone Persistence Unit. Users can package a set of entities along with persistence.xml in an OSGi bundle and deploy it. The standalone persistence units can then be accessed in either Java SE or EE mode by other hybrid applications.

EE resources as OSGi services:
Standard EE resources like JDBC DataSource, JavaMail resource, JMS resource will be available in OSGi service registry. Only globally available resources of aforementioned types will be exposed as OSGi services. Global resources can be deployed either via administration API or by use of EE APIs like @DataSourceDefinition or @Resource. A resource is globally available if it is bound to global JNDI namespace.

For CDI/OSGi integration, refer to CDI team's specification.

For OSGi/JAX-RS integration, refer to Jersey team's proposal.

META-INF/services, OSGi and Java EE APIs:
Refer to [4]

c) Embeddable GlassFish:
We will allow GlassFish to be embedded in existing OSGi as well as non-OSGi runtime and yet function like a regular GlassFish.

d) Generic OSGi features:
OBR integration: We plan to have OBR integration in place so that the GlassFish deployment process can resolve dependencies of bundles via OBR. OBR integration will also allow a la carte access to GlassFish modules which will be a compelling feature for the embedded use case. Decision to have our own OBR is not yet taken. In this release, we plan to provide OBR support so that users can use their own repository if they want to.

Apache Gogo shell: This provides a scripting environment for

administering the OSGi runtime.

OSGi Web Console: A web frontend for administering the OSGi runtime.

4.2. Bug/RFE Number(s):

4.3. In Scope:

4.4. Out of Scope:

4.5. Interfaces:

4.5.1 Exported Interfaces

Interface: org.osgi.framework, version=4.2
Stability: Standard
Comments: Applications deployed as OSGi bundles can use this API.

Interface: org.osgi.service.http, version=4.1
Stability: Standard
Comments: For use by OSGi applications

Interface: Embedded GlassFish API
Stability: Evolving

Interface: Enterprise OSGi Spec
Stability: Standard

4.5.2 Imported interfaces

Interface: org.apache.felix
Stability: Evolving
Exporting Project: Apache Felix (http://felix.apache.org)

Interface: org.apache.felix.bundlerepository
Stability: Evolving
Comments: OBR integration

4.5.3 Other interfaces (Optional)

4.6. Doc Impact:

New chapter needed in developer guide documenting how to develop OSGi enabled Java EE applications. GlassFish administration document needs to be updated with new CLI and GUI functionality.

4.7. Admin/Config Impact:

"asadmin deploy" command has already been extended with type=osgi in V3 to support deployment of OSGi bundles. It has to be extended to support distribution of bundles remote instances in a clustered environment.

The executable shell will be based on Apache Felix Gogo, which is an implementation of OSGi RFC 147, which is a proposed standard shell for OSGi environments. The shell is more advanced and supports expected shell features (e.g., scripting, piping, variables) as well as more advanced features (e.g., closures, dynamic method invocation). We will not implement any command like "asadmin osgi-shell" in this release. Users will have to use native interface of the OSGi shell.

Felix web console is already usable in GlassFish V3 to administer OSGi runtime. We shall customize it and make it part of GlassFish update centre if not part of default distribution. If resources are available, we shall integrate it with our admin console, otherwise it will be accessible as a separate URL.

4.8. HA Impact:

XXX: Need further discussion.

4.9. I18N/L10N Impact:
New resource bundles will be introduced which need to be localized as per GlassFish requirement. The localized content will be made available as just like any other localized content in GlassFish v3.

4.10. Packaging & Delivery:
Most of the functionality will be implemented in separate OSGi bundles so that we can decide to make them part of default distribution or make them available via update centre at a later point of time.

4.11. Security Impact:

4.12. Compatibility Impact
For various enterprise OSGi features, we need to be compatible with the corresponding specs.

4.13. Dependencies:

5. Reference Documents:

1. OSGi in GlassFish V3
http://wiki.glassfish.java.net/attach/V3FunctionalSpecs/GFv3-OSGi-onepager-v0.3.txt

2. Modularisation of GlassFish using OSGi
http://wiki.glassfish.java.net/attach/V3FunctionalSpecs/GFv3Prelude-OSGi-onepager-v0.2.txt

3. GlassFish/OSGi Wiki Home:
http://wiki.glassfish.java.net/Wiki.jsp?page=OSGi

4. JDK's Service Provider Mechanism & OSGi
http://wiki.glassfish.java.net/Wiki.jsp?page=JdkSpiOsgi

6. Schedule:

6.1. Projected Availability:
Same schedule as GlassFish v3.1