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: heavy@sun.com

    1.3. Date of This Document:
        06 Jan 2009

2. Project Summary
    2.1. Project Description:
        In this document, we describe OSGi [1] related features supported in GlassFish application server version 3.0. This is a continuation of our effort which began during GlassFish V3 Prelude release. Please refer to our earlier proposal [2] for a background to the problem.

    2.2. Risks and Assumptions:
        Some of the features proposed here are not standardised anywhere. e.g., we propose to allow Java EE applications to be packaged and deployed as OSGi bundles. Since this is the first time any Java EE platform is implementing such an extension, we don't yet know all the challenges involved. SO, we may not be able to provide all kinds of features of the Java EE platforms to OSGi enabled Java EE applications. The assumption here is that supporting OSGi enabled Java EE applications is not a violation of any of the Java EE platform rules. We would like to mention here that we will continue to support standard compliant, non-OSGi Java EE applications in the server without any change in behavior.

3. Problem Summary
    3.1. Problem Area:
        As stated in our earlier proposal [2], OSGi module system is used to implement the primary features of GlassFish V3 release, such as: modularity, extensibility, and embeddability. Since OSGi framework is the kernel of the application server, there is a need to have better administration support for the framework.

        Today's enterprise Java applications have a growing need for sophisticated dependency management, better life cycle management and a dynamic service platform. Hence applications would like to use OSGi to meet some of these requirements.

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

4. Technical Description:
     4.1. Details:
        The features proposed in this document fall into following categories:
        a) Administration of OSGi runtime:
        For the former one, we shall enhance GlassFish CLI command sets and admin console. It will allow users to install, uninstall OSGi bundles, browse currently installed bundles, query their status and manage their lifecycle. More details on this is available in section #4.7.

        b) Support for OSGi enabled Java EE applications:
        As discussed in the prelude one-pager [2], Java EE applications typically run outside of OSGi context. Going forward, we shall allow Java EE applications to be deployed as OSGi bundles so that at runtime they have OSGi BundleContext available to them. To begin with, we shall not automatically wrap Java EE applications as OSGi bundles although

such wrapping is possible in most cases, but we shall expect user to package their application as OSGi bundle. The existing deploy command will be modified to accept such artifacts. Such a hybrid application can leverage functionality provided by OSGi as well as Java EE platform. Existing Java EE archive formats except "EAR" format fit nicely with OSGi bundle format. So, user just has to add OSGi metadata in their Java EE archive to convert into an OSGi bundle. Please refer to OSGi specification for details about OSGi manifest headers. For bundles packaged as .war or .rar files, it is easy to figure out the Java EE application type, but an application packaged as .jar file must contain the following proprietary manifest header to indicate that it is a Java EE application type:
GlassFish-Application-Type: EJB
An OSGi enables Java EE application can even use Export-Package header to export some packages to be used by other deployed applications. Please note this is different from the encapsulation model of ordinary Java EE applications.

c) Implementation of standard OSGi services:
We shall implement the following standard OSGi services:
OSGi HTTP Service: This is described at [4].
Transactions in OSGi (OSGi EEG RFC 0098): This is nothing but Java Transaction API with the added requirement that UserTransaction, TransactionManager and TransactionSynchronizationRegistry objects be made available in OSGi service registry under javax.transaction.UserTransaction, javax.transaction.TransactionManager and javax.transaction.TransactionSynchronisationRegistry names respectively.
d) Exporting Java EE services to OSGi service registry:
Applications can choose to export the following services to OSGi service registry:
EJBs, Resources (JDBC DataSource, JavaMail resource, JMS resource) and JPA EntityManagerFactories. Although we have not decided yet, but we are thinking of having an annotation which programmers can use to export EJBs to service registry. We shall enhance the admin commands and console to allow users to export resources to service registry.

4.2. Bug/RFE Number(s):
https://glassfish.dev.java.net/issues/show_bug.cgi?id=6848
http://forums.java.net/jive/thread.jspa?messageID=320807
http://forums.java.net/jive/thread.jspa?messageID=318199

4.3. In Scope:

4.4. Out of Scope:
Patching/Upgrading at runtime is not going to be implemented in this release. We will not automatically wrap non-OSGi libraries installed in library directories as OSGi bundles. We will not automatically convert non-OSGi applications to OSGi applications. We will not provide any tools to package Java EE applications as OSGi bundles. Users should use existing tools like bnd [6] to achieve the same. bnd can be used from command line as well as from Ant and Maven.
We will not support an ear file to be deployed as an OSGi bundle in this release.

4.5. Interfaces:
4.5.1 Exported Interfaces
Interface: org.osgi.framework, version=4.0
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

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

Interface: org.glassfish.api
Stability: Evolving

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:
We will introduce the following CLIs:
(id refers to the unique bundle id assigned by OSGi framework to every bundle. It is a strictly increasing number persistence across framework restart.)

```
install-bundle <URL>              # Install a bundle
uninstall-bundle <id>             # Uninstall the bundle
list-bundles                      # List all installed bundles
list-headers <id> [<id>...]       # Display header of given bundle
resolve-bundle <id> [<id>...]     # Resolve bundle(s)
start-bundle <id> [<id>...]       # Start bundle(s)
stop-bundle <id> [<id>...]        # Stop bundle(s)
packages <id> [<id>...]           # List exported packages
```

We will provide commands to export resources to OSGi service registry. It will be achieved either by extending existing commands or adding new commands. It is not decided yet.

We will also enhance the implementation of deploy command to accept OSGi enabled applications.

All the above tasks should be possible using the Admin Console as well. All of them require a running server.

We also need support from config module. A newly installed bundle may need to add default configuration information in domain.xml. So, we depend on "configuration pluggability" feature in GlassFish.
[TODO: insert link to config pluggability one-pager]

4.8. HA Impact:
We need support for distribution of OSGi bundles from DAS to remote instances when such bundles are installed using "asadmin install-bundle" command with a target specified as either a cluster or a remote instance.

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 OSGi bundle fragments 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.

4.11. Security Impact:

In the developer profile of GlassFish, we would like to enable
Felix remote shell [3] which accepts connections over telnet protocol.
It can be configured to bind to 127.0.0.1 only so that it accepts
connections from the host only reducing the security vulnerability.

By default, bundles installed via install-bundle command
have same privileges as any other bundle in glassfish/modules dir.
Bundles deployed as applications will be less privileged. e.g., they
would not be allowed to override or export APIs that are part of the
Java EE platform. Similarly, they will not be allowed to export
standard services like TransactionManager. [TODO: discuss with Richard]

4.12. Compatibility Impact
NONE

4.13. Dependencies:
1. Felix to support Java class transformation in order to run
JPA in OSGi context.

2. Configuration Pluggability support in GlassFish.

5. Reference Documents:

1. OSGi Module System:
http://osgi.org

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

3. Felix Remote Shell:
http://felix.apache.org/site/apache-felix-remote-shell.html

4. OSGi HTTP Service:
http://www.osgi.org/javadoc/r4v41/org/osgi/service/http/HttpService.html
5. Apache Felix project:
http://felix.apache.org

6. Bnd Bundle Tool:
http://www.aqute.biz/Code/Bnd

7. Providing HK2 services on OSGi:
http://wikihome.sfbay.sun.com/asarch/attach/Attachments%2FHK2OSGi.pdf

6. Schedule:
6.1. Projected Availability:
Same schedule as GlassFish v3.0