// Delete comments before final submission.
// make sure that the response fits within 80 columns
01234567890123456789012345678901234567890123456789012345678901234567890123456789
1. Introduction
    1.1. Project/Component Working Name:
        Supporting Addons in Appserver EE.

    1.2. Name(s) and e-mail address of Document Author(s)/Supplier:
        Binod. P.G : binod.pg@sun.com

    1.3. Date of This Document:
        16/10/06

2. Project Summary
    2.1. Project Description:

        The current addon infrastructure provided by appserver is only sufficient
        to install and configure an addon on top of appserver PE. This project
        aims at extending that capability to appserver EE also.

    2.2. Risks and Assumptions:

        Adoption of this functionality by other products is not controlled by this
        project. That introduces a risk of finding a bug very late in the release
        cycle.

3. Problem Summary
    3.1. Problem Area:

        Addon functionality is limited to appserver PE.

    3.2. Justification:

        EE version of appserver is open sourced and is likely to be the base for
        Java EE SDK. It is important to have infrastructure capabilities in appserver
        to support Addon products that get installed on top of appserver in Java EE
        SDK.

4. Technical Description:
     4.1. Details:

     1.0 Current Design

An addon is expected to follow the packaging guidelines as explained in the document below.
[http://wikihome.sfbay.sun.com/blueprints/Wiki.jsp?page=PackagingAndIntegration].
An addon is expected implement an installer plugin and a configurator plugin. Installer plugin will get invoked during the SDK installation.
Location of Appserver installation will be provided to inatller plugin.
Installer Plugin is expected to install the common bits for the addon as well as copy the configurator plugin to the lib/addons directory.

Configurator plugin is invoked prior to starting of the appserver so that it can modify the domain.xml to include the required entries and modify the server.policy. Addon configurator plugin directly modifies the domain.xml today. Offline AMX, when it is ready, can be used for this purpose.

2.0 Cluster / Remote Instance Support for Addons

2.1 Installation

Installer plugin will be invoked during installation in DAS machine. The Installed addon will be synchronized to different instances when node agents starts up.

[Open question: If the nodeagent process is used to do the synchronization, then, it might be running with a different user-id. It might not have permission to access the required directories. If we cant assume that the same user is running the nodeagent, it would be required for the user to explicitley install the addon in the remote machines]

Some addons deploy system applications (war, ear etc.) to the appserver
by copying the application to the autodeploy directory. However these applications will just behave as the user applications. Instead of copying
to autodeploy directory, they should be copied to lib/install/applications
directory. Currently all the appserver system applications reside in this directory. By default, the applications in this directory will be considered of type "system-all" and will be deployed in all instances (cluster/standalone/das) of the appserver.

2.2 Configuration

Configurator Plugin will be invoked at three places in the server startup.

2.2.1 Domain Startup

The plugin can modify the domain.xml and insert common entries for lifecycle module. Currently lifecycle-module does not support object-type. So, it cannot be flagged as "system-all" "system-admin" etc. We will add this support. If addon want one particular system application to have a prticular type, it should modify the domain.xml during domain startup.

2.2.2 Prio to Start Instance In DAS

When start-instance command is executed in DAS, the configurator plugin will be executed. At this time, domain.xml can be modified to add any system-property element. Typically this may be used to configure ports opened by the instances.

2.2.3 Prior to Actual Start Instance in the Remote Instance

The configurator plugin can create any instance specific directory (eg INSTANCE_ROOT/jbi) at this time.

2.3 Enhancement to the interfaces

An explicit interface for Installer and Configurator will be provided to addon implementations that use this feature. Javadoc for theese interfaces are available at

http://www.glassfishwiki.org/gfwiki/attach/OnePagersOrFunctionalSpecs/addons.zip

```
jbi-bpel_installer.jar
   |- META-INF/services/com.sun.appserv.addons.Installer
   |- JBIInstallerImpl.class (implements Installer)
   |- InstallerUtil.class
   |- InstallerMore.class
   |- jbi-bpel.jar
      |- META-INF/services/com.sun.appserv.addons.Configurator
      |- JBIBPELConfiguratorImpl.class (implements Configurator)
      |- Another.class
      |- More.class
      |- Util.class
```

3.0 Uninstallation Support

```
     -------------------------

     3.1 Unconfiguring the domain and instances.
     ----------------------------------------
     Currently the registry(DOMAIN_DIR/config/addon.properties) supports
only
     enable/disable. This will be enhanced to add configure/unconfigure
support.

     eg: opensso.jar.configured=false

     During the restart of DAS or instance, if the above entry is found in
the
     registry, configurator plugin will be invoked to unconfigure the
addon.
     Registry file will be synchronized to INSTANCE_ROOT/config during the
     instance startup.

     The configurator plugin of the addon is expected to rollback the
changes
     it made during the configuration.

     3.2 Uninstalling the addon.
     -------------------------
     During SDK uninstallation, the installer plugin will be executed to
uninstall
     the addon. This step will remove all the installation time activities
like
     removing the addon's installation directory and system applications
copied
     to the $INSTALL_ROOT/lib/install/applications directory.

     4.2. Bug/RFE Number(s):
        // List any Bug(s)/RFE(s) which will be addressed by this proposed
change.
        // Provide links to the Issue tracker Bug(s)/RFE(s)where possible

     4.3. In Scope:
        // Aspects that are in scope of this proposal if not obvious from
above.

     4.4. Out of Scope:
       CLI and GUI support for the project is not in scope.

     4.5. Interfaces:

        4.5.1 Exported Interfaces

            Interface: Addon Installer Plugin
            Stability: Unstable.
```

Former Stability (if changing):
Comments:

Interface: Addon Configurator Plugin
Stability: Unstable.
Former Stability (if changing):
Comments:

4.5.2 Imported interfaces
NA

4.5.3 Other interfaces (Optional)
NA

4.6. Doc Impact:
The new feature need to be docuemented in the developer's guide.

4.7. Admin/Config Impact:
No impact on CLI and GUI.

4.8. HA Impact:
No specific impact.

4.9. I18N/L10N Impact:
No impact.

4.10. Packaging & Delivery:
No impact.

4.11. Security Impact:
No impact.

4.12. Compatibility Impact

Currently, it is possible to execute the main class of the addon installer using "java -jar <xxx.jar". Since the current approach remove the need for the main class, addon installer is expected to provide a main method, where it accepts the required configuration. This is the only way to keep the backward compatibility for someone to add a new addon after installing appserver.

4.13. Dependencies:

5. Reference Documents:

6. Schedule:
6.1. Projected Availability:
// Dates in appropriate precision (quarters, years)