# Asynchronous Release

## Definition of Asynchronous Release

**Asynchronous Release**: Asynchronous Release is change in the DSpace release process and version numbering process on modules within the DSpace trunk to allow more flexibility adding prebuilt Addon modules into DSpace.

## Goals

The primary goal of Async release is to break the the authoritative grip that dspace-parent has on the version assignment and dependencyManagement in the DSpace trunk modules such that:

1. Make the build in the DSpace Modules Source Tree dependent on a specific dspace-api and dspace-xmlui-api versions. Specifically, dspace-statistics and dspace-discovery
2. Make a packaged release of DSpace that includes a combination of Trunk and Module projects.
3. More easily package releases of DSpace cyclically while allowing minor updates of core modules to occur more often, to easily provide a minor update path for DSpace.
4. Allow immediate consumption of minor maintenance releases of DSpace modules without requiring major new releases and marketing or reliance on SNAPSHOT builds.

## Requirements

1. dspace-parent is removed from the project, all dependency management is maintained in the modules that are using those dependencies, a dependency management section may be added to dspace/pom.xml or dspace/modules/pom.xml to support overriding the default dependencies identified in dspace-api, dspace,xmlui-api etc.
2. dspace-api should be released with a separate version number separate the general DSpace 1.x.x release. For instance, taking on the idea of using Single digit version numbers for releases, We would promote using 1.8.0 for a release of dspace-api while using 8.0 for the release. (*dspace-api-1.8.0.jar* would be in *dspace-release-8.0*)
3. dspace-api should be separated into individual modules. The codebase should be evaluated to determine the extraction points, But strong candidates for pushing into separate packages are:
    - **dspace-core/dspace-core-api:** containing Interfaces for Domain Model and Services centered on *org.dspace.content, org.dspace.plugin* and *org.dspace.core (stored outside DSpace trunk)*
    - **dspace-core/dspace-core-impl:** containing implementations of DSpace above services,*org.dspace.plugin* and *org.dspace.core, org.dspace.content and org.dspace.browse (until the circular dependency on Browse can be removed)*
    - **dspace-core/dspace-legacy-storage**: containing *org.dspace.storage.rdbms* and *org.dspace.storage.bitstore*
    - **dspace-core/dspace-legacy-app**: Applications and addon support found in *org.dspace.apps*
    - **dspace-core/dspace-cli**: A refactored ScriptLauncher that uses Service Manager to get Commands.

## Changes to DSpace Data Model

We will want to introduce an API over the DSpace Object Model that will allow us acquire DSpaceObjects from DSpaceObjectServices without depending on dspace-api.  Once in place, there would be two features of the API and Implementation that benefit Addon projects:

1. DSpace Addons would only need to reference the Data Model Interfaces and the Service Interfaces
2. DSpace Trunk would define an implementation of these Services that adhered to the Service and Data Model API.

Benefits: We will then be able to support only releasing the API when it changes and addons which depened ont eh API would not need to be rereleased over new versions of DSpace

New module Projects

- modules/dspace-legacy-api
    - Data Model
        - BitstreamInterface
        - ItemInterface
        - CollectionInterface
        - CommunityInterface
        - SiteInterface
        - ...
    - Services
        - LegacyStorageService
        - LegacyDSpaceObjectProvider
- dspace/dspace-core/legacy-services-impl
    - TODO

## Changes to Maven Multimodules Build

dspace/trunk/pom.xml (http://scm.dspace.org/svn/repo/dspace/trunk/pom.xml)

Ceases to exist, we will move all projects that use dspace-parent to use dspace-pom (http://scm.dspace.org/svn/repo/modules/dspace-pom/trunk/pom.xml) which is released separately from trunk.

### trunk/dspace/pom.xml

Continues to fill its role as an assembly point for constructing a DSpace instance. Important changes are that the assembly is changed so that dependencies are not defined here and a separate "cli" project is created to hold dependencies that will be assembled into the DSpace lib directory. This allows a specialized location to add customizations to go specifically into the lib directory codebase.  Upgrades to DSpace will require migrating your changes in dspace/pom.xml dependencies into the cli project instead.

```
modules
   pom.xml
   cli
      pom.xml
      src
          main
              java
              resources
```

### trunk/dspace/modules/pom.xml

Will be defined as the place that dependencyManagement is customized for a specific DSpace instance.  Dependency Management is then moved out of the scope of *dspace-parent*. This allows individual DSpace instances to mediate the versions of dependencies chosen for a specific deployment.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.dspace</groupId>
    <artifactId>modules</artifactId>
    <version>11-SNAPSHOT</version>
    <packaging>pom</packaging>

    <name>DSpace Addon Modules</name>
    <url>http://www.dspace.org</url>
    <description>DSpace Addon Modules</description>

    <parent>
        <artifactId>dspace-pom</artifactId>
        <groupId>org.dspace</groupId>
        <version>8</version>
    </parent>

    <modules>
        <module>cli</module>
        <module>xmlui</module>
        <module>lni</module>
        <module>oai</module>
        <module>jspui</module>
        <module>sword</module>
        <module>solr</module>
    </modules>

    <dependencyManagement>
        <dependencies>
            <!-- DSpace core and endorsed Addons -->
            <dependency>
                <groupId>org.dspace</groupId>
                <artifactId>dspace-api</artifactId>
                <version>2.1-SNAPSHOT</version>
            </dependency>
```

## Possible Changes to DSpace Module versioning scheme

| Module | Scheme | 1.7.0 | 1.8.0 | 1.9.0 | |
|---|---|---|---|---|---|
| dspace | N | 10 | 11 | 12 | once a year |
| dspace/modules | N | 10 | 11 | 12 | once a year |
| dspace /modules/... | N | 10 | 11 | 12 | once a year |
| dspace-api | M.M | 1.7.0 | 2.0 | 3.0 | Increments as needed |
| dspace-jspui | M.M | 1.7.0 | 2.0 | 2.1 | Increments as needed |
| dspace-jspui-api | M.M | 1.7.0 | 2.0 | 2.0 | Increments as needed |
| dspace-xxx-yyy | M.M | 1.7.0 | 2.M | M.M | Increments as needed |
| | | | | | |

M.M: First Bit is Major release number, Second is minor, Major releases are reserved for API, Configuration and DB changes, minor releases are for bug fixes and other backward compatible release changes.

## Next Steps

- Push a patch for the prototype into the JIRA for DSpace and get feedback from the committers group on the refactorings.

## Important Caveats in Maven Build

1. Assembly and Dependency Management cannot currently occur in the same Maven module, we currently dealt with this by having dspace-parent do dependencyManagement and the dspace assembly project do the assembly.  Approach needs to be retained in moving dependencyManagerment into the dspace/modules/pom.xml and continuing to reserve assembly as part of dspace/pom.xml. This can be solved by using dependencyManagement "imports" rather than inheritance to define the dependency versions critical to the project.