RIFF Submission Service
Technical and User Guide

21 December 2007

Table of Contents

1. Purpose

The purpose of this document is to provide an overview of the RIFF Submission Service package along with technical details of the services implemented. It also provides some answers to typical FAQs and a guide to configuration of the service.

2. Overview

Note:The RIFF Submission Service (RIFF SS)is not a software package for end users. It does not come packaged with a friendly user interface. It is a framework for job scheduling and job chaining and should only be used by developers or system administrators.

The RIFF SS provides a service-oriented framework to support the packaging and routing of content and metadata from a source application to a target repository. The RIFF SS delivers a job chaining framework and in an APSR context also includes default implementations of the successful RIFF workflow projects in varying degrees. Specifically the default workflows are journals (from Open Journal Systems (OJS) to DSpace and Fedora Repositories), conference papers (from Open Conference Systems (OCS) to DSpace), image collections (from iSpheres to DSpace) and Word Processing Documents (from Scholar's Workbench (SWB) to DSpace).

The RIFF SS distribution comprises a number of default services including submission, status reporting and job management. From a software perspective the goals of the RIFF SS development were:

- to make the Submission Service lightweight but flexible;
- not to be a burden for IT support staff (and other developers) to maintain;
- to use common and stable technologies;
- to be platform-independent.

The software was written in Java and XSLT ensuring it is platform-independent. It is also repository independent and can be made to work with any repository software and does not have to be deployed alongside either the content-generating or repository application.

Interoperability is achieved partly through the use of the Australian METS Profile as a content format and partly through the ability to create custom tasks within a submission. Furthermore the service can operate in various contexts, thus enabling deployment in a variety of scenarios. For this reason, no assumptions have been made about the authentication environment for deployment so it is up to the installer to place protections around the various services or improve the code to meet individual requirements.

The core of the distribution is the open source Quartz scheduling software (http://www.opensymphony.com/quartz/). The RIFF SS acts as a wrapper around Quartz and imposes some rules for the configuration of submission processors (jobs).

3. Services

Service: **submit**

Parameters: **type** or **package** and **repository**. In addition any parameters from the jobs.xml file requiring to be overridden by the request. Refer to the Job Configuration section for more information. **Type** is repository and package conjoined, for example repository=DSpace and package=OJS is equivalent to type=DSpaceOJS indicating the type of job being submitted is an OJS package that needs to be deposited in a DSpace repository.

Description: The submit service is responsible for accepting a request to submit a job type to the scheduler. A job creator class is instantiated by the submit service to create a scheduled job based on parameters in the submit request combined with those stored in the jobs.xml file for that particular job type.

Service: **status**

Parameters: **name** to show a single job with name=**name**; no parameters show all jobs currently scheduled. Specifying **html=true** results in an HTML view of the status page else a raw XML response is returned.

Description: The status service shows the current status of jobs scheduled within the RIFF SS. This service could be used for example to redirect to a page rendering the status servlet response after a submission.

Service: **manage**

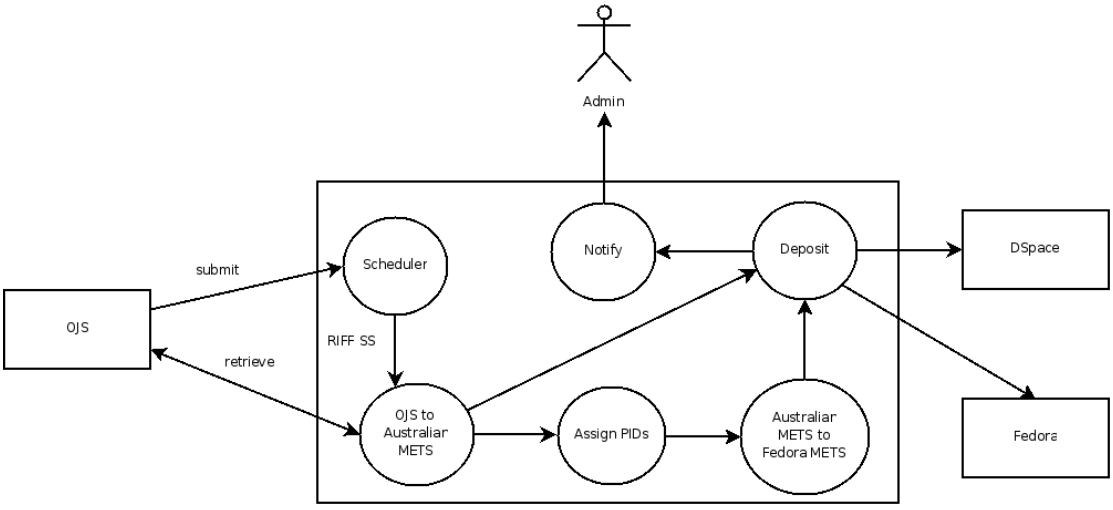
Parameters: **action** to indicate what management service is being used; **reschedule** and **delete** are currently supported, each requiring a **name** parameter as well. In addition, **action=reschedule** requires a **cron** parameter indicating the new scheduled time or an empty string if to run immediately. Note that the assumption is a system admin would be the only user to have access to these services, there is no checking on the job's owner for these services in the current implementation and would need to be added.

Description: The manage service is for administrative services. Currently supported are enabling rescheduling of a job and deletion of a job from the scheduler. There are security issues associated with status and manage services so need to be deployed with these in mind.

4. Default Workflows

A diagrammatical overview of the workflows provided in the default distribution is shown below. All are similar and all use Australian METS as their packaging format.

OJS Journals



OCS Conferences

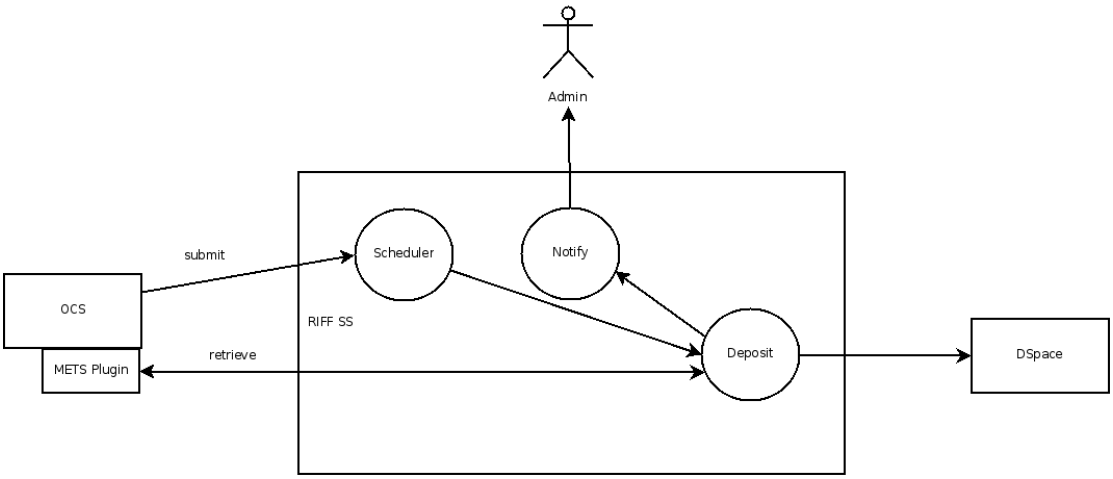
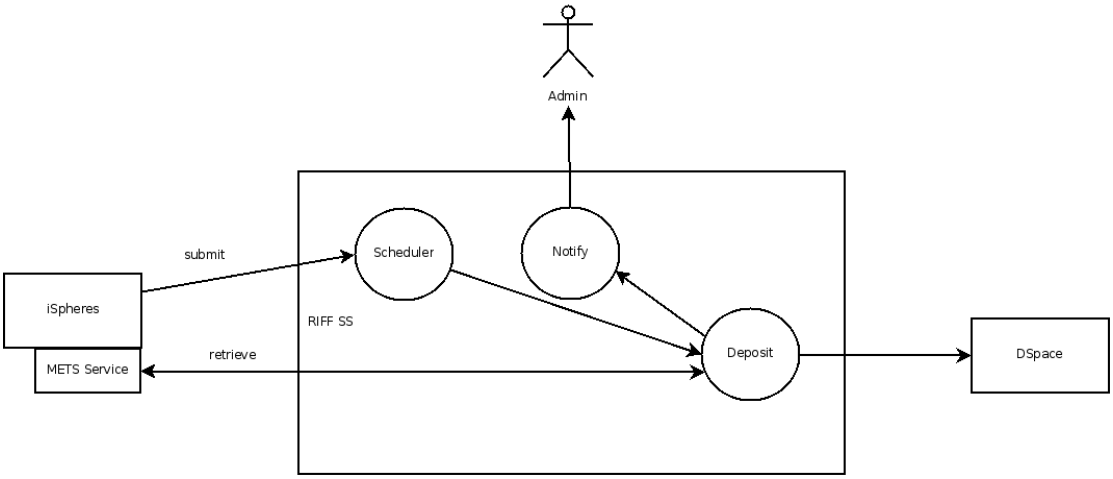
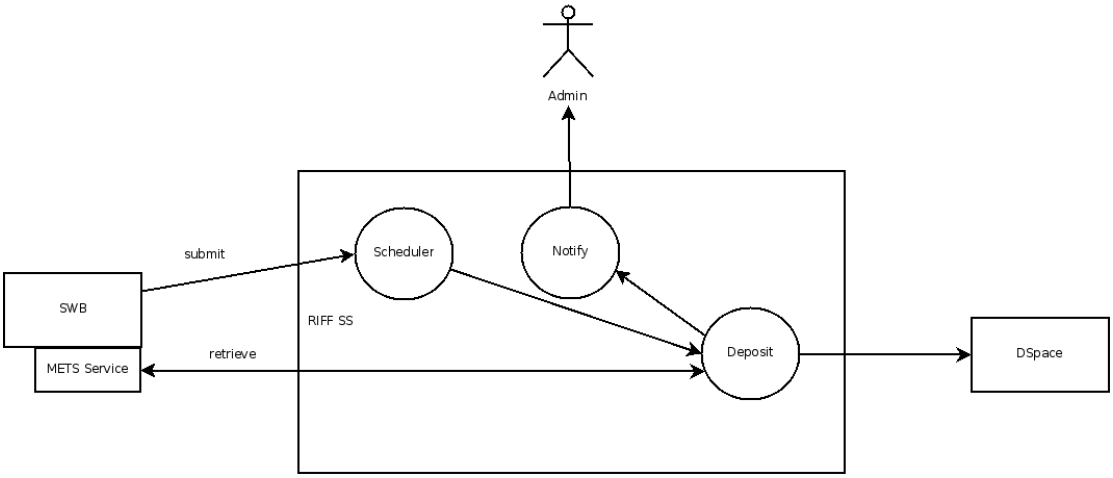


Image Collections (iSpheres)



Documents (Scholar's Workbench)



5. Job Configuration

Jobs are defined in the jobs.xml file. An example job defining a DSpace OJS Journal job type is shown below with comments interspersed to explain each element.

```
<!-- root element is jobs -->
<jobs>
  <!-- the workdir is a directory where working files and directories are created for all the jobs in the RIFF SS. The RIFF SS webapp must have read/write access to this area and it should not be accessible outside the webapp -->
  <workdir>/usr/local/apps/apsr/riffs3/workdir</workdir>
  <!-- Each type, or class, of job must be defined. A job type is made up of a content type and repository type. In the example below an OJS-to-DSpace job is being defined, thus the root element of this job is DspaceOJS. Multiple jobs of this type can be submitted with different names and different parameters, this job definition just provides a template for a “standard” job of this type -->
  <DSpaceOJS>
    <!-- The creator element provides for definition of a class responsible for the initialisation and scheduling of a DSpaceOJS job -->
    <creator>
      <class>au.edu.apsr.riffs3.jobcreator.DefaultJobCreator</class>
    </creator>
    <!-- the tasks element is a wrapper around all tasks which make up this job type. By default tasks are chained in sequence -->
    <tasks>
      <!-- all jobs require a name and the name must be unique across the entire RIFF SS. The name would typically be passed as a parameter when calling the submit service. -->
      <name>DSpaceOJS</name>
      <!-- Each task comprises a name (not unique), a class, and an arbitrary set of parameters. All or any of the parameters can be overridden by passing an alternative value to the submit service, however all parameters must be defined in the jobs.xml else they will be ignored if using the DefaultJobCreator -->
      <task>
        <!-- The name of a task, unlike the name of a job, does not have to be unique -->
        <name>OJS-to-AustMETS Transform</name>
        <!-- The class responsible for carrying out the Job. The class should implement the StatefulJob interface as defined in Quartz -->
        <class>au.edu.apsr.riffs3.job.TransformJob</class>
        <!-- An arbitrary list of parameters can be defined for each task, with or without values. Any parameter can be overridden by the submit service if the caller passes through the parameter with an alternative value. Parameters are simply key/value pairs. An argument list can also be included in a parameter definition which may be useful (as in the example below). -->
        <param>
          <key>stylesheet</key>
          <value>/usr/local/apps/apsr/riffs3/stylesheets/ojs2austmets.xsl</value
```

```
>
            <args>
              <key>agent-organisation</key>
              <value>The Australian National University</value>
              <key>original-format</key>
              <value>text/html</value>
            </args>
          </param>
          <param>
            <key>source</key>
          </param>
          <param>
            <key>target</key>
            <value>austmets.xml</value>
          </param>
```
<!-- The trigger is used to specify the scheduled time. A trigger can also be passed via the submit service by specifying a “cron” parameter with or without (for one-off immediate execution) a value. The format of the trigger must conform to that set out in the Quartz documentation -->
```
          <trigger>
            <cron>0 15 10 15 * ?</cron>
          </trigger>
        </task>
        <task>
          <name>DSpaceRemotePackager</name>
          <class>au.edu.apsr.riffs3.dspace.job.RemotePackagerJob</class>
          <param>
            <key>repository-service-url</key>
            <value>http://localhost:8080/dspace/packager</value>
          </param>
          <param>
            <key>riff-base-url</key>
            <value>http://localhost:8080/riffs3</value>
          </param>
          <param>
            <key>source</key>
            <value>austmets.xml</value>
          </param>
          <param>
            <key>eperson</key>
            <value>me@home</value>
          </param>
          <param>
            <key>collection</key>
          </param>
          <param>
            <key>community</key>
```

```
            <value>123456789/1</value>
          </param>
          <param>
            <key>originalMimetype</key>
            <value>text/html</value>
          </param>
          <param>
            <key>type</key>
            <value>AustJ</value>
          </param>
        </task>
        <task>
          <class>au.edu.apsr.riffs3.job.NotificationJob</class>
          <name>Notification</name>
          <param>
            <key>notify</key>
            <value>me@home</value>
          </param>
          <param>
            <key>mailserver</key>
            <value>mymailhost.au</value>
          </param>
          <param>
            <key>type</key>
            <value>completion</value>
          </param>
        </task>
      </tasks>
    </DSpaceOJS>
</jobs>
```

6. Source Code Guide

A brief overview of the source organisation follows. For further details individual code modules should be examined. A javadoc option with the build file is available (use ant target of 'javadoc').

riffs3->config

jobs.xml: the job configuration file where jobs are defined and task sequences are specified. This is parsed by Apache Commons Configuration.

log4j.properties: the logging configuration file. Logging for the application uses the Apache log4j pacakage.

quartz.properties: configuration of the Quartz scheduler.

riffs3->docs

Install guide, license and the like can be found in the docs area.

riffs3->etc

initial setup of the Quartz database and the RIFF SS web.xml.

riffs3->lib

All external jars and associated license details.

riffs3->src

Java Source Code

riffs3->stylesheets

XSLT Source Code

riffs3->src->...->dspace

All modules specific to DSpace.

riffs3->src->...->job

All classes implementing the StatefulJob interface. Each class equates to a RIFF SS task.

riffs3->src->...->jobcreator

All classes responsible for initialising and scheduling Quartz jobs. A default creator class is included with the distribution (described later)

riffs3->src->...->listener

All classes responsible for performing some action on activation of a trigger. By default two listeners are implemented: one for single run jobs which cleans up after a job is run and remove all trace from the system; and another for recurring jobs which cleans up working files. Both also perform some rudimentary exception handling.

riffs3->src->...->servlet

Classes implementing an available service.

riffs3->src->...->util

Classes containing methods useful to numerous other classes and XSL Java extension methods.

Available separately are DSpace packager modules and Manakin Theme Editor which comprise the RIFF package and are available on the APSR Web Site.