EPrints Hybrid Storage - Management and Configuration

Table of Contents

1 Introduction

EPrints 3.2 introduces an abstracted storage layer which provides the ability for data hosting solutions such as Amazon S3 to be utilised as a storage back end to EPrints. The advantage of this is that you can "plug-in" to multiple storage services at the same time and control these with a local "Storage Policy".

In this tutorial we look at the some of the storage interfaces that EPrints can use, and also how to modify the storage policies to suit the needs of a modern repository.

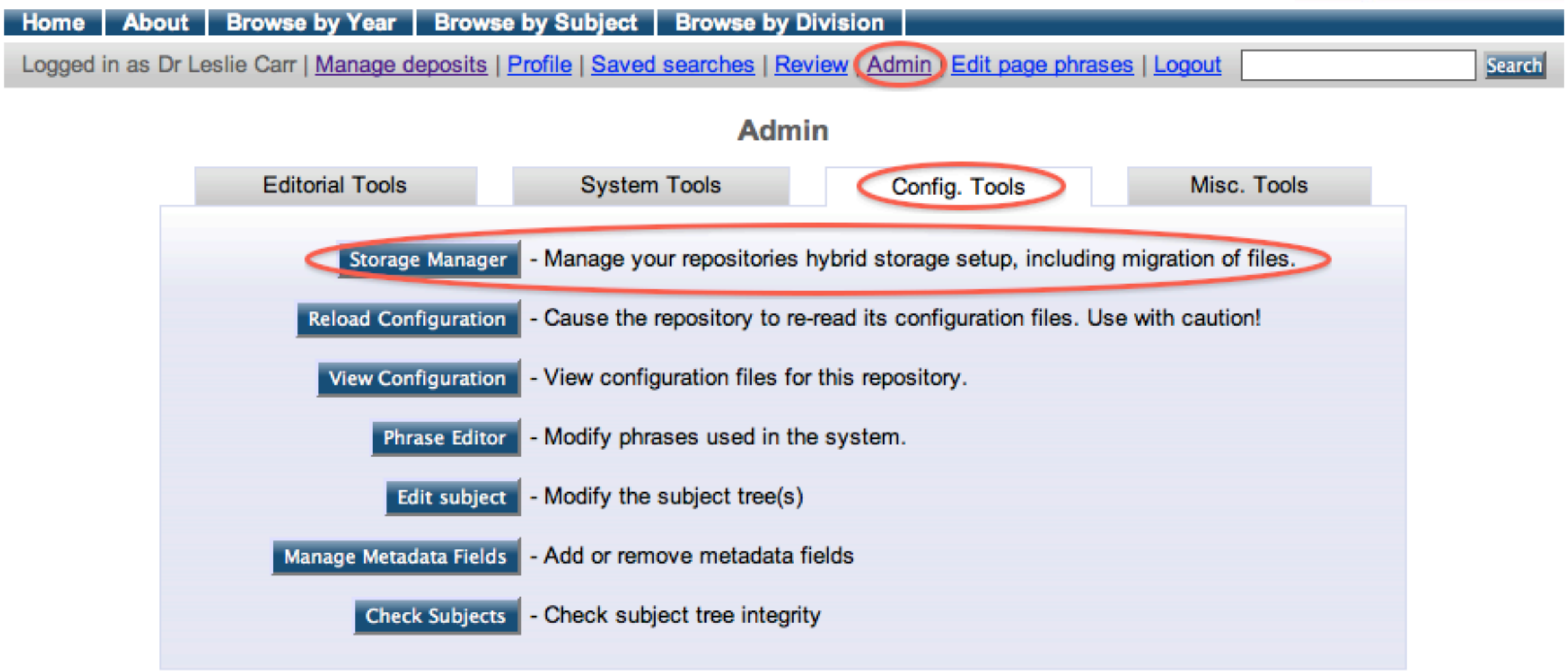
2 Powerful Storage

The EPrints team have been working on a number of storage plug-ins which are available both through files.eprints.org and will be released in EPrints 3.2. In this section we introduce a few of them:

Name	Plugin ID	Description
Local	Local	Saves to the local hard disk.
Local Compressed	LocalCompress	A useful little addition to the local plug-in that compresses files transparently (no change is seen in the user interface). Good for saving local disk space and archiving old and rarely used objects.
Sun Honeycomb	HoneyComb	Supports Sun's Honeycomb platform (no longer commercially available). This plug-in supports the API used by this class of highly robust archival storage.
Amazon S3	AmazonS3	Plug-in to support the most widely known cloud storage provider. As well as supporting basic storage functionality the plug-in also supports Amazon Cloudfront for direct localised delivery of resources.
Sun Cloud Storage	SunCSS	Very similar to the Amazon S3 offerings. Note that this is a beta services and EPrints is a testing partner.

3 Viewing your storage service usages

The Storage Manager screen can be found under the **Config Tools** tab of the admin screen.



Home | About | Browse by Year | Browse by Subject | Browse by Division
Logged in as Dr Leslie Carr | Manage deposits | Profile | Saved searches | Review | Admin | Edit page phrases | Logout
Search
Admin
Editorial Tools
System Tools
Config. Tools
Misc. Tools

Storage Manager - Manage your repositories hybrid storage setup, including migration of files.

Reload Configuration - Cause the repository to re-read its configuration files. Use with caution!

View Configuration - View configuration files for this repository.

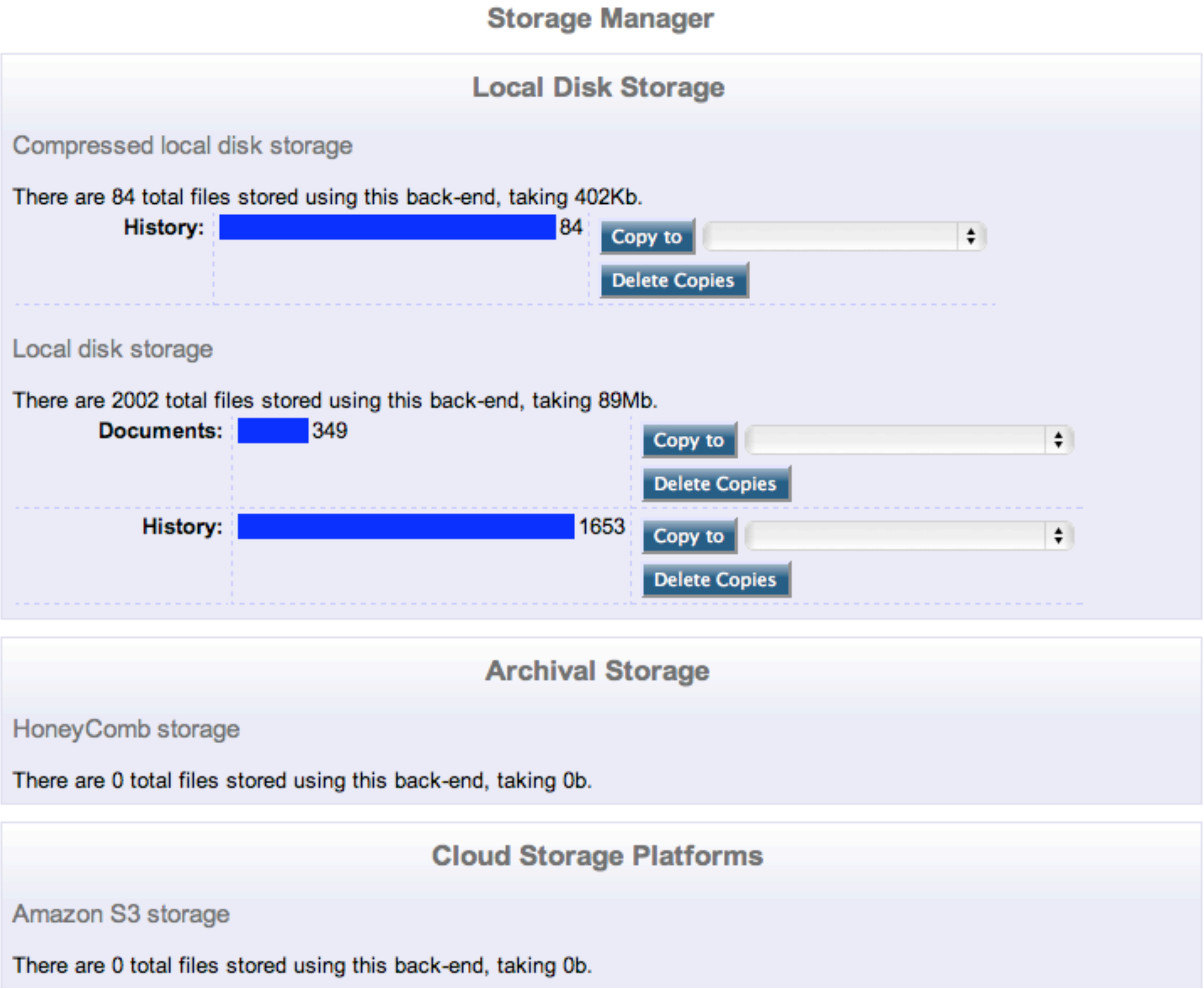
Phrase Editor - Modify phrases used in the system.

Edit subject - Modify the subject tree(s)

Manage Metadata Fields - Add or remove metadata fields

Check Subjects - Check subject tree integrity

The figure below shows the Storage Manager. From this screen you can easily view where your objects are and how much space they consume. You can also move them between storage platforms with a single click.



Storage Manager

Local Disk Storage

Compressed local disk storage

There are 84 total files stored using this back-end, taking 402Kb.

History: 84

Local disk storage

There are 2002 total files stored using this back-end, taking 89Mb.

Documents: 349

History: 1653

Archival Storage

HoneyComb storage

There are 0 total files stored using this back-end, taking 0b.

Cloud Storage Platforms

Amazon S3 storage

There are 0 total files stored using this back-end, taking 0b.

4 Managing your Storage Policy (Exercises)

4.1 View / Edit the Storage Policy

The EPrints Storage Controller is managed by a policy defined in an xml file. This config file (storage/default.xml) can be edited by clicking the **View Configuration** button available from the **Config Tools** tab of the admin interface. **storage/default.xml** is located near the bottom of the available list of files. By clicking on this file you can view and edit it in your browser.

4.2 Understanding the default Storage Policy

Like many configuration files in EPrints, the storage policy is defined in xml using the EPrints Control language/namespace (epc) to define decisions in an XSLT like fashion.

Below you can see a copy of the current default storage policy. The annotations should help explain what each line does. Note that by default, all files are stored locally.

```
<store>                                              Start Policies
  <epc:choose>                                       Begin Choice Section
       <epc:when test="datasetid = 'document'">      If current object is document
          <plugin name="Local"/>                     Use Local plug-in
       </epc:when>                                   End document condition
       <epc:otherwise>                               Otherwise
          <plugin name="Local"/>                     Use Local plug-in
     </epc:otherwise>                                End otherwise condition
  </epc:choose>                                      End Choice Section
</store>                                             End Policies
```

4.3 Exercise 1: Volatile Files

A good starting point for managing storage services is to decide what happens to volatile files. These are files that are generated by the repository for internal use (e.g. image previews). As such it is unlikely that these files will need to be stored off site or preserved.

Volatile files are part of the document dataset and we can differentiate these from other files by looking for a relation which exists between the two types of files.

Edit the default storage policy and insert the following code to handle volatile and non-volatile document files differently.   The code replaces the ‘<plugin name=”Local” />’ line inside the ‘epc:when’ section of the code.

```
<epc:choose>
  <epc:when test="$parent{relation_type} = 'http://eprints.org/relation/isVolatileVersionOf'">
     <plugin name="Local"/>
  </epc:when>
  <epc:otherwise>
     <plugin name="LocalCompress"/>
  </epc:otherwise>
</epc:choose>
```

After changing the policy you will need to add a new EPrint to the repository. Upload a PDF, JPEG or GIF to the EPrint, then view the Storage Manager screen to verify that there are files in more than one location.

4.4 Exercise 2: Multiple Storage Locations

With the above done, it should now be easy to add a second location to store non-volatile documents in. This can be done by simply adding a new <plugin> tag to the relevant section.

EPrints handles multiple storage locations for both storage and delivery by simply processing them in the order they appear in the storage policy. For storage, files are stored in all locations. For delivery, the file is served from the first location in the config file. In the event that this is not available, it moves onto the second, and so on.

4.5 Exercise 3: Storage policy based upon repository metadata

In this section you will use the epc language to access file metadata. This metatdata will then be used to control the storage of the item.

Each object we handle with the storage controller is a "file" object in eprints and thus the following pieces of metadata are just a few that are directly available:

- filename - Name of the file
- mime_type - Type of the file (e.g. PDF, JPEG, etc...)
- filesize - The size of the file

A conditional can be inserted into the policy file to use this data to make decisions:

```
<epc:when test="mime_type = 'application/pdf'">
  <plugin name="StoragePluginName"/>
</epc:when>
```

Add the above rule to store PDF files (application/pdf) in a different location from everything else stored in the repository.

Note that EPrints and User metadata is also available. It is possible to make a decision based on (for example) who uploaded the item or which subject area the publication record is associated with.

4.6 Example Solution

The following code will solve all exercises contained in this document.

```
<store xmlns="http://eprints.org/ep3/storage" xmlns:epc="http://eprints.org/ep3/control">
  <epc:choose>
    <epc:when test="datasetid = 'document'">
      <epc:choose>
        <epc:when test="$parent{relation_type} = 'http://eprints.org/relation/isVolatileVersionOf'">
          <plugin name="Local"/>
        </epc:when>
        <epc:otherwise>
          <epc:choose>
            <epc:when test="mime_type = 'application/pdf'">
              <plugin name="SunCSS"/>
              <plugin name="LocalCompress"/>
            </epc:when>
            <epc:otherwise>
              <plugin name="Local"/>
            </epc:otherwise>
          </epc:choose>
        </epc:otherwise>
      </epc:choose>
    </epc:when>
    <epc:otherwise>
      <plugin name="Local"/>
    </epc:otherwise>
  </epc:choose>
</store>
```