Research360

EPrints Integration with the Hitachi Content Platform

June 2012

1. Aim

The University of Bath has recently purchased a Hitachi Content Platform (HCP) as a next-generation storage solution. This is the candidate system to be used to store archived research data and the ability to create and retrieve information from the HCP is therefore a crucial part of the research data management plans of the University.

The University of Bath already uses EPrints as its repository, Opus, for research papers. EPrints was therefore selected for use in a proof-of-concept integration with the HCP to determine whether the HCP could be used to manage archived research data. The data archive would thus present an interface with which repository managers are already familiar and would not add any significant new support requirements.

2. Technical Solution

2.1 HCP Configuration

As a proprietary system, the HCP cannot be directly modified to meet the requirements of the EPrints storage API. However, it already has most of the properties that are needed and is extensively configurable.

The architecture of the HCP is hierarchical, with the HCP as a whole being divisible into a large number of “tenants”, each of which can be further subdivided into “namespaces”. Each namespace is part of a tenant, which is part of the whole, and at each level the components can be individually configured, the default configuration being mostly inherited from the relevant parent component.

Part of the configuration of each tenant and/or namespace is the object management and retention policy. A variety of possible policies can be used, one of which is to set the namespaces as a Write Once Read Many (WORM) drive, meaning that stored objects can be created but not then modified or deleted. This could be used for an archive setting, but was not used as the configuration for the development system. A decision on whether WORM configuration should be used on a production system has yet to be taken. Namespaces and tenants have usage quotas associated with them that can be amended at will, with the restriction that the quota cannot be reduced to less than the current usage. These quotas place maxima on the number of namespaces in a tenant, the number of stored items, or the disk space used by the stored items.

For the purposes of the pilot installation, a single tenant named “research_data” was created, which contained three namespaces with self-explanatory functions: “dev”, “test” and “live”. Initially, each namespace was given a small 5 GB usage quota.

Each tenant and/or namespace can have user accounts associated with it. These accounts can be used for management through the HCP's management console interface or for access to material (subject to access restrictions at each level). The console interface does not give access to the objects themselves, but allows configuration changes and monitoring of performance and usage of the tenant/namespace. For the purposes of the pilot installation, a single username and password was set up for management of the tenant, and another username and password was set up for the REST interface to use for access to each of the research_data namespaces. The credentials for this user need to appear in the configuration for the developed module. An alternative architecture, in which each user at the University would have an associated identity on the HCP, was rejected because of the requirement for large scale identity management on the HCP, which is not designed for this

purpose, and because it would require complicated mechanisms in the EPrints module for the appropriate user credentials to be derived for access to the storage system.

The REST interface to the HCP is relatively simple: It uses URLs of the form https://namespace.tenant.global_domain_name/directory_structure/filename to access a location; create a file (with HTTP *PUT*); download a file (with HTTP *GET*); or, if possible, delete a file (with HTTP *DELETE*). It is also possible to set up chunked access to a large file, enabling the file to be downloaded in several pieces of more easily handled size. However, this was not explored for the purposes of the pilot.

When a file is created, the directory structure is created if it does not already exist. One slight complication in the interface arises from the use of self-signed certificates in the HCP, which are also assigned to the global domain name. Certificate checking therefore needs to be turned off as an HTTP client will decide that they are invalid for access to an individual namespace. More awkwardly, authentication is done solely through encrypted cookies, rather than using HTTP Basic authentication, which means that it is not possible to carry out testing using a normal web browser.

Use of the REST interface gives a certain degree of flexibility in the future, as EPrints is not the only software which can be extended to access the HCP in this manner (but see below).

2.2 EPrints

EPrints also has a hierarchical structure, with each installation being made up of one or more 'repositories' that can have independent configuration of many options. This includes flexible storage, which allows the use of custom Perl modules for access to storage media that are not supported out of the box. Thus storage can be configured for use for the whole of an EPrints installation, or for individual repositories within this installation, or for specific media types. For the pilot system, a custom module was written and configured to be used in the whole of a test installation of EPrints. It should be noted that the EPrints Wiki documentation on the storage API differs from the API as implemented in the Local module distributed with the software. The latter was used as the model for development rather than the documentation.

Determining exactly how items in EPrints will relate to those in the HCP is a policy decision for the repository managers. The developed module allows four possible approaches to the deletion of items. This is possible because storage modules do not handle the deletion of information about an item from the EPrints database, and therefore neither its appearance nor removal from the EPrints interface. The four settings for deletion of items in the HCP via the module are:

1. Always report failure without contacting the HCP, so that both EPrints and the HCP appear to be WORM archives;
2. Always report success without contacting the HCP, so that items appear to be removed from EPrints while the HCP is treated as a WORM archive regardless of its configuration;
3. Ask the HCP to delete the item and report success or failure as received in the HCP's response to the request, so that EPrints and the HCP appear to have the same configuration (default);
4. Ask the HCP to delete the item and always report success, so that items can be removed from the EPrints interface while being retained in the HCP. This means that

access can be removed from objects that cannot actually be deleted because of the HCP settings.

2.3 Development Approach

The staged approach taken to development was to first ensure access to the HCP, using the Linux *curl* command with appropriate options to upload a file and then download it again. These *curl* commands appear in redacted form as comments in the HCP module. A Perl script was then written to upload and download a file using the *LWP* module. This was chosen rather than the simpler *HTTP::Request* and *HTTP::Response* modules because the latter modules did not easily handle the cookie authentication required by the HCP. Finally, the Perl code was adapted to become a module which implemented the EPrints storage API. Each stage was tested by uploading and then downloading a small graphic file (Figure 1).

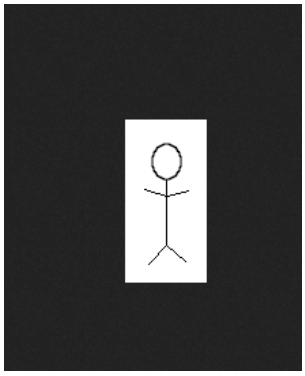

Figure 1: Small graphic file used for testing during module development.

The screenshots in Figures 2 to 5 show the pilot system in use for the deposit and retrieval of the small graphic file (Figure 1). Apart from use of the HCP module, the only customisation to the EPrints installation that was been made was to set up HTTPS access: the point of this demonstration is that the process appears no different to the user than storage to the local file system on the EPrints server.

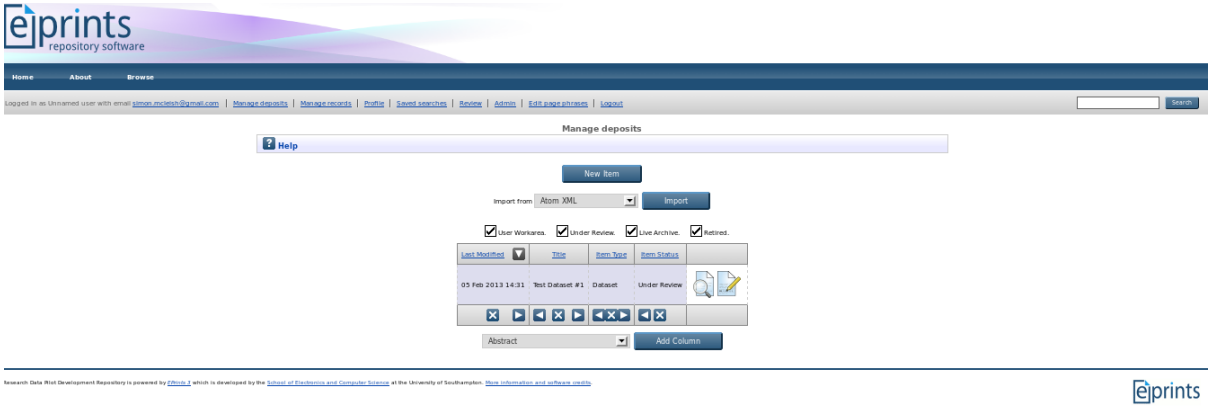

Figure 2: Click on ‘New Item’. *Screenshot of EPrints, which is made available under a GPLv3 licence.*

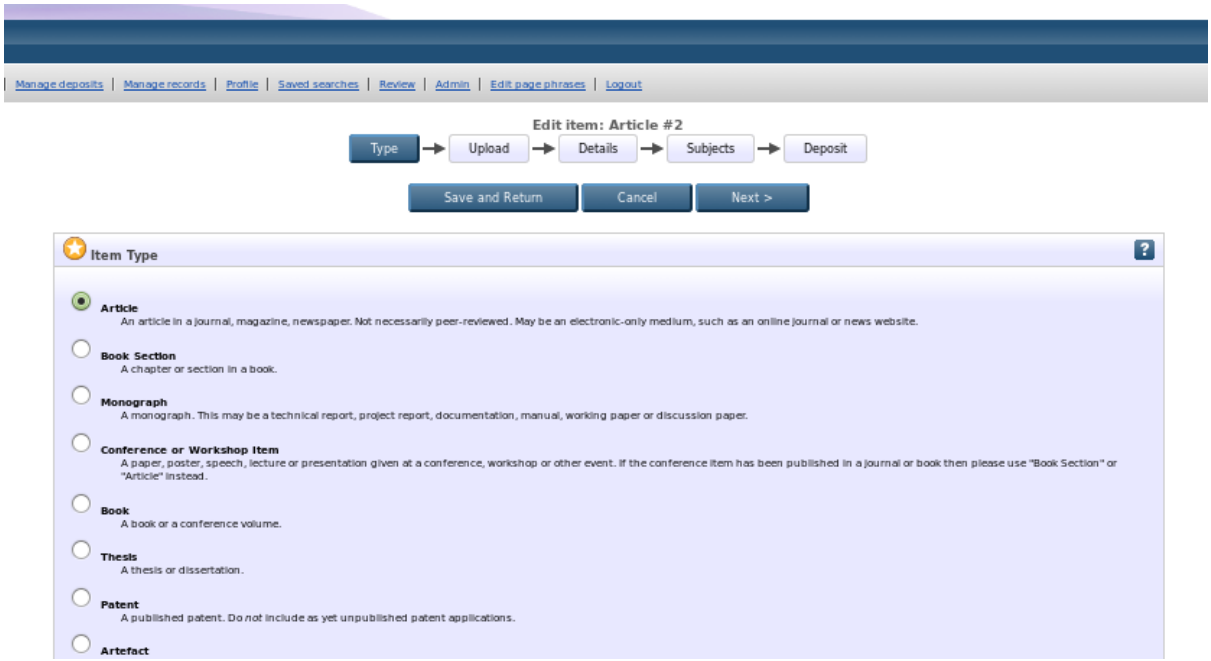

Figure 3: Select Dataset then click on 'Next'. *Screenshot of EPrints, which is made available under a GPLv3 licence.*

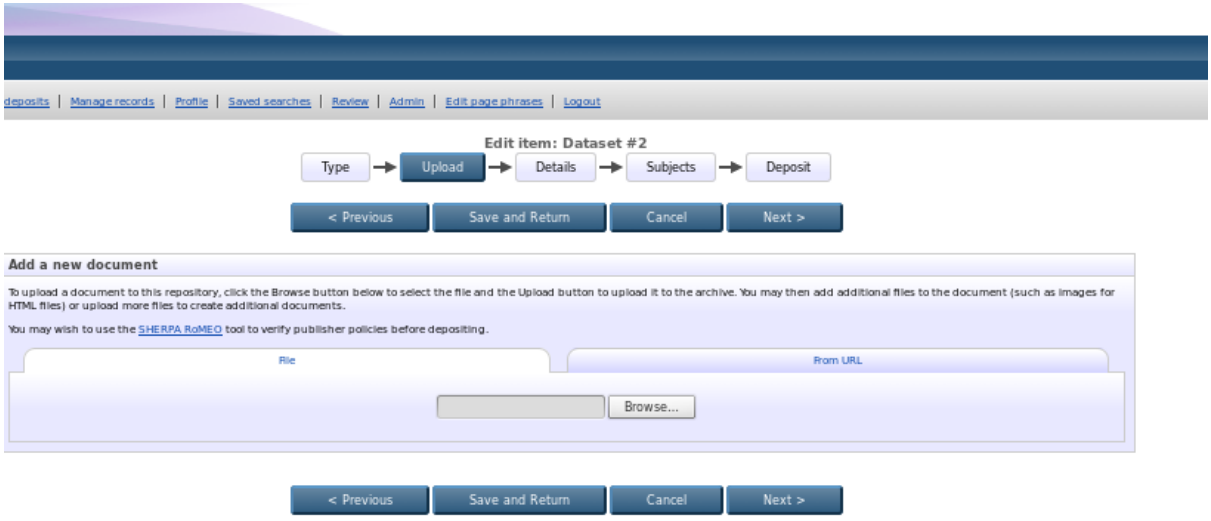

Figure 4: Click on 'Browse', find a file and the click 'Open'. *Screenshot of EPrints, which is made available under a GPLv3 licence.*

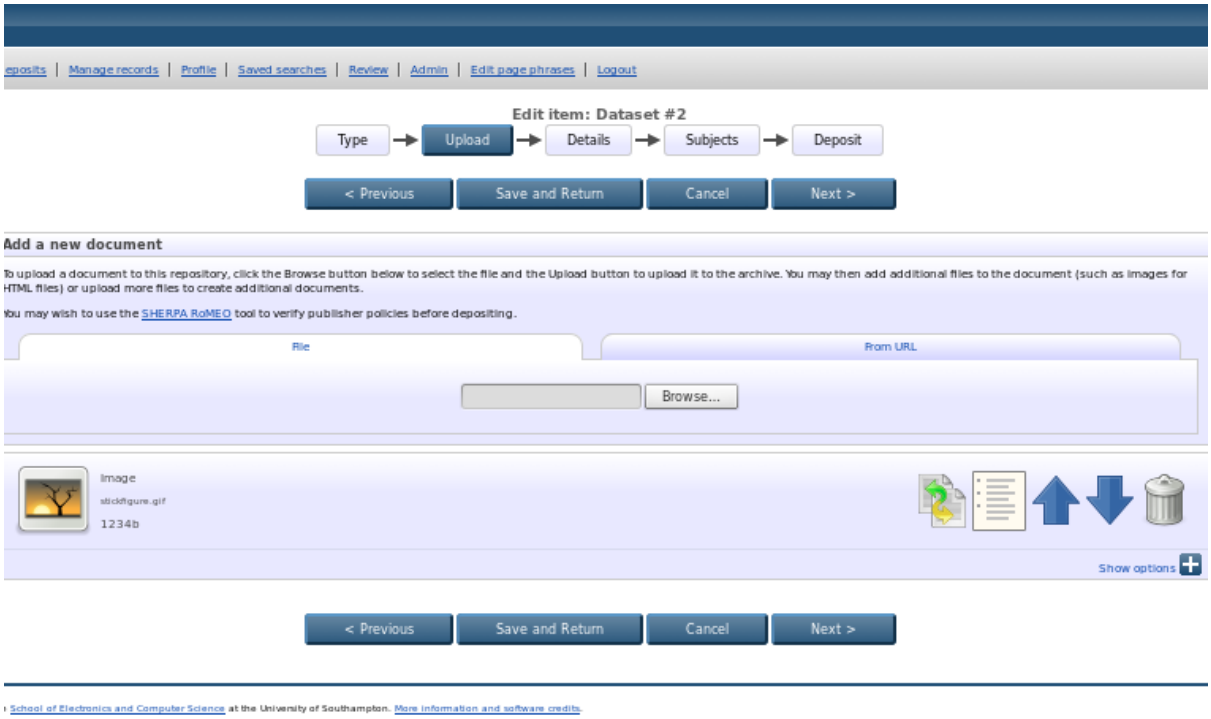

Figure 5: Rather than go through the complete process, just click on the icon for the new image. *Screenshot of EPrints, which is made available under a GPLv3 licence.*

3. Difficulties Encountered

The use of cookies by the HCP and the way that it inhibits the use of a standard web browser for testing has already been noted. Another problem encountered during testing was web browser caching when the EPrints interface was used, which led to some confusion in the testing of the last stage of development. In addition, the method used by the HCP interface to record the space used, which includes the HCP system metadata as well as the actual content, was found to be confusing when seeking to check that a new file has been uploaded.

The API allows the use of more than one approach to the communication between the module and EPrints. However, it was difficult to find detailed descriptions of the storage API on the EPrints Wiki, meaning that some experimentation was necessary before finding a solution that correctly managed both uploading and downloading and which worked with the requirements of the REST interface.

4. Moving to a Production System

Several steps are required to move from this pilot module to a pilot production system. This section considers these and gives some recommendations as to how they might be implemented.

4.1 HCP Configuration

As previously indicated, the pilot HCP configuration sets a low quota for the three namespaces. In production use, the quota for at least the production namespace will need to be increased to some agreed and sensible values, and processes put in place for monitoring usage (for example, ensuring that quota alerts are sent to the repository manager and/or the HCP maintainer in computing services).

The HCP can support a confusing array of object management configurations, with options for WORM storage, read/write storage, automatic deletion, automatic passage between WORM and read/write storage, object versioning, and deletion with shredding (i.e. making the deleted object unrecoverable). All of these options can be configured globally, on a per-tenant basis, or on a per-namespace basis. As previously explained, the EPrints module developed supports four possible modes for deletion that keep the EPrints and the HCP deletion of the object either independent or coupled to a degree.

The HCP also supports other University of Bath activities, and thus it is strongly recommended that configuration changes are made on the namespace and tenant level only, so that other production systems are not affected.

4.2 EPrints Configuration and Support

The previously noted configurability of EPrints hybrid storage suggests that a production model for an institutional data repository could make use of this feature. The existing Opus repository is supported as a production service through the EPrints team based at the University of Southampton. In the future, it would be possible to use local storage for Opus and then modify it to support storage on the HCP via REST, with the HCP module being used for storage for a separate research data repository added to the existing installation. With regard to future support, it is not expected that the HCP module would be especially complex to support, unless either the storage API for EPrints is modified for a future release or Hitachi alter the REST API for the HCP. Additionally, it might be desirable to separate the workflows for research data and research papers, something which could be configured on a per-archive basis.

Supporting chunked transfer to and from the HCP should be relatively trivial to add if required, as the form of the developed module would potentially support this with minimal modification. However, it is not clear whether the chunked nature of the transfer can be passed on to the user's browser by EPrints. Clarification should be sought on this issue if it is considered to be useful. Additionally, the REST Developer's Guide for the HCP warns that “[c]hunked transfers impose a modest performance penalty relative to normal PUT requests. You should carefully consider your application architecture before using this feature.”

4.3 Statistical Data on Usage

Statistical information about usage is available from the HCP through the REST interface, or through the tenant management console, although the latter is a less flexible method to obtain the data. Note, however, that this only provides information about the storage of data e.g. number of objects, total size of objects, rather than access. Whilst the former is likely to be of use to estimate future storage requirements, the latter will be required for reporting to the University and external funders.

Statistical information in EPrints is not provided as standard, but can be gathered in a variety of ways including the IRStats add-on package. Such a solution would make it possible to gather information about the frequency of access requests for individual archived data objects, provided that EPrints is used as the interface to download them. In the event that other systems, such as a virtual research environment or a CRIS, might eventually be extended to enable direct access of objects stored on the HCP tenant of the data repository, then any statistics obtained from EPrints would need to be combined with statistics from these other systems to give a true picture of the access to research data.

4.4 Limitations of the architecture

The main limitation of using REST for access to the HCP is speed. For even very small files, object creation and access take a noticeable period of time, most of which is spent in the establishment of the connection to the HCP. The relative connection time will be reduced for larger files, where the actual transportation of the data takes up a larger proportion of the total download time than the establishment of the connection: a 3GB file will transfer faster in KB/s than a 3KB file. The user perception of the speed of access is something that should be monitored, though there are limitations to how much can be done to improve the speed due to the inherent nature of the connection to the HCP.

The HCP module uses the standard EPrints directory/file naming structure. This uses numerical directory names ordered by the date on which a file was stored rather than the date on which it becomes accessible, the latter being dependent on deposit workflow considerations and potentially being infinitely later. The file name of an object stored on the HCP cannot therefore be easily guessed by applications other than EPrints, although it could be decoded using the appropriate database table from EPrints.

5. Availability of Developed Components

The module and associated technical documentation will be deposited in the University of Bath's Opus repository under a GNU GPL 3.0 licence, with a copy also submitted to EPrints files (e.g. http://files.eprints.org/777/). The Research360 blog (blogs.bath.ac.uk/research360) will provide supporting publicity. The work will consist of:

- A README file with installation instructions (See section 6: Documentation);
- A sample configuration file (with no real life operational information);
- The HCP module itself;
- A copy of or link to this document.

It is anticipated that the module itself could be used to support development of new modules that access REST interfaces for other storage solutions. Note, however, that it is unlikely to be directly transferable without modification due to the use of cookies in the HCP API.

6. Documentation

The contents of the README file, which accompanies the module, are duplicated below:

How to Install

1. Set up EPrints, ideally using SSL. This should also set up an EPrints userID whose home directory is the EPrints installation location (`$EPRINTS_HOME`, in this document - which can be omitted if it's your current directory when running through these instructions). The Perl libraries needed should already be installed either as part of the EPrints installation or in the core Perl installation.
2. If LWP::Protocol::https is not installed, then install it. If you do not do this, then the following error will appear in the apache error log:

   ```
   Can't locate object method "ssl_opts" via package "LWP::UserAgent".
   ```

3. Set up a tenant and namespace to use for storage from the archive on the HCP. The module will set up the same directory structure as EPrints natively uses to manage its files on the remote machine within this namespace. A user needs to be created that can

access this namespace using the REST API, and then hashed versions of the credentials used by this user need to be calculated, which are used to authenticate by the module. The UNIX command to create the hashed credentials is

```
echo hcp-ns-auth=`echo -n username | base64`:`echo -n password | md5sum`
 | awk '{print $1}'
```

replacing username and password with the appropriate values.

4. Unzip the archive in a convenient location (e.g. `/tmp`). This is most conveniently done as the EPrints user, because otherwise the remaining tasks will also require re-setting file ownerships as root.
5. As EPrints, copy HCP.pm to `$EPRINTS_HOME/perl_lib/EPrints/Plugin/Storage/HCP.pm`. Then change the group ownership of the file to `www-data (chown eprints:www-data $EPRINTS_HOME/perl_lib/EPrints/Plugin/Storage/HCP.pm)` to match the other files in the directory.
6. Modify `$EPRINTS_HOME/lib/storage/default.xml` and configure storage mechanism to be "HCP" for the archive or mime types which should use the HCP.
7. Modify HCP.pl to match local HCP installation. The information needed is the URL to access the namespace, and the access credentials. The deletion mode for the module should also be set. There are four options:

   0 Always report failure (i.e. do not attempt to delete files from the archive); items will not disappear from either the HCP or EPrints - this means that both EPrints and the HCP are treated as WORM archives.

   1 Always report success (also not attempting to delete files from the archive); items will disappear from the EPrints interface but not from the HCP, so that EPrints is a read/write archive while the HCP is treated as a WORM archive.

   2 (Default) Make a deletion request to the HCP and report the result; how this behaves will depend on the deletion settings for the file on the HCP, and it will be removed from EPrints precisely when it has successfully been removed from the HCP, so that the deletion settings of the HCP will be "replicated" in the EPrints interface.

   3 Make a deletion request to the HCP but always report success; items will always disappear from the EPrints interface but may not be deleted from the HCP, thus enabling access to be removed to items which the HCP will not delete.

8. As EPrints, copy the modified HCP.pl to `$EPRINTS_HOME/archives/[archive-id]/cfg/cfg.d/HCP.pl`, replacing [archive-id] with the archive ID to be used; a copy will need to be placed in each archive's configuration where it is desired to use the HCP for storage (the HCP.pl files can be customised differently for each archive).
9. As root, restart apache. (This may not be necessary, but it doesn't hurt.) If you want to see more information about the activities of the HCP module in the logs, then change line

77 of HCP.pm to

```
$self->{debug} = 1;
```

before restarting apache.