Building an institutional repository with Xythos
# Table of contents

Introduction ...................................................................................................................................... 1  
Conventions and references........................................................................................................ 1  
Business requirements .................................................................................................................... 1  
What is an institutional repository? .............................................................................................. 1  
The challenge ............................................................................................................................... 2  
The Xythos toolkit ............................................................................................................................ 2  
WebDAV ...................................................................................................................................... 2  
Categorization .............................................................................................................................. 3  
Information lifecycle management ............................................................................................... 3  
E-mail archiving and external content.......................................................................................... 3  
Access control lists ....................................................................................................................... 3  
Tickets .......................................................................................................................................... 3  
APIs for customization and integration ........................................................................................ 4  
LDAP integration.......................................................................................................................... 4  
Portal deployment ........................................................................................................................ 4  
Import/export ................................................................................................................................ 4  
Scanning integration ..................................................................................................................... 4  
Putting it all together .................................................................................................................... 5  
Scenario #1: Loose integration .................................................................................................... 5  
  Security..................................................................................................................................... 5  
  Metadata................................................................................................................................... 6  
  Records management .............................................................................................................. 7  
  Pros and cons ......................................................................................................................... 7  
  Summary .................................................................................................................................. 7  
Scenario #2: Tight integration ...................................................................................................... 8  
  Transfer and accession ............................................................................................................ 9  
  Security................................................................................................................................... 10  
  Metadata.................................................................................................................................. 10  
  Pros and cons......................................................................................................................... 10  
  Summary ................................................................................................................................ 11  
Scenario #3: One system.............................................................................................................. 12  
  Pros and cons......................................................................................................................... 13  
  Summary ................................................................................................................................ 13  
Future directions ............................................................................................................................ 14  
Conclusion ..................................................................................................................................... 14  
Appendices .................................................................................................................................... 15  
Appendix A: Simple DSpace example ....................................................................................... 15  
Appendix B: Complete DSpace example ................................................................................... 17
Introduction

In higher education, an institutional repository is the ultimate home of published materials. In other words, the institutional repository can be the home of much of the university’s important intellectual property, such as research publications, historical documents, masters and doctoral theses, course syllabi and notes, and recorded lectures.

Since this content is nearly all stored in files, the Xythos server seems like a logical component of an institutional repository, if not the institutional repository itself. This article is designed to answer some important questions about using the Xythos server in this capacity, such as…

1. What part of the requirements for institutional repositories can the Xythos server address?
2. How can the Xythos server integrate with institutional repository applications?
3. Are there any Xythos features in particular that are important for institutional repositories?
4. How might desktop tools contribute to the institutional repository?
5. Which flavor of the Xythos server (Enterprise Document Manager, Digital Locker, or WebFile Server) is most appropriate?

Conventions and references

This white paper uses the term “Xythos server” broadly to refer to all three server products, the Enterprise Document Manager, the Digital Locker, and the WebFile Server. All three might have a place in your institutional repository strategy. Your use case will indicate which version of the product is the best fit.

Throughout this white paper, you will see links to more complete descriptions of the technologies and customer examples discussed. If you still have questions, contact your Xythos representative, or post your question at http://developer.xythos.com.

Business requirements

What is an institutional repository?

Features and functions

At both a technical and business level, an institutional repository is a relatively simple application. It is a sub-species of a digital library, an electronic archive of content that users can search and browse. Several years ago, when knowledge management was all the rage, digital libraries were the quintessential knowledge management application.

Users access content by searching and browsing by metadata (author, title, subject, publication date, etc.). Consequently, contributors have to provide this information when adding content to the institutional repository. This requirement creates one of the biggest human obstacles to the success of the institutional repository: the average person doesn’t want to take the time to learn the institutional repository application, and then fill out the metadata for every single item added to it. Therefore, many institutional repositories provide bulk upload mechanisms, to leverage information that may already be provided with the content.

The people who manage the institutional repository need to exercise some quality control. At the same time, they also need to stay out of the way of the publication process, if their intervention
isn’t strictly needed. Quality control prior to publication narrows the flow of newly published content; quality control after publication makes it harder to identify and correct problems.

To summarize, an institutional repository has the following capabilities, ranked according to their importance:

- **Showstopper**
  - Browse, metadata
  - Search, metadata
  - Contribution, manual
  - Metadata, basic (author, title, etc.)
  - Metadata, standard (Dublin Core, Open Archive Initiative, etc.)
  - QA after publication

- **Must have**
  - Search, content
  - Contribution, automated ("harvesting")
  - QA before publication
  - Security, single sign-on
  - Security, content-level access control

If your system doesn’t have the “must have” features, you can still call the application an institutional repository. However, the “must have” features represent some of the make-or-break features on which the success of the institutional repository depends.

**The challenge**

Technologically, psychologically, and organizationally, the institutional repository faces a dilemma. On the one hand, everyone wants it to succeed as a catalog of important content. On the other hand, responsibility for maintaining it falls to a small group of specialists, such as university librarians, who become the experts at using the specialized institutional repository application. Even the best-intentioned users may not have the time or technical skills to contribute to the institutional repository, especially since there are specialists already responsible for it.

The success of the institutional repository therefore hinges on broadening its scope. More users need to easily add content to it. Other systems that contain the content to be catalogued in the institutional repository need to be connected to it. Once there’s enough content in the institutional repository, and a large number of people are contributing to it, the project has reached the critical mass needed for success.

**The Xythos toolkit**

Before explaining how to use Xythos technology to address these requirements, it’s important to understand what the technology can do. Here is a summary of the pieces of the Xythos toolkit that might be used in an institutional repository project. Which pieces get used, and in what fashion, depends on the integration approach, discussed in a later portion of this white paper.

If you’re already familiar with Xythos technology, please feel free to skip this section.

**WebDAV**

The Xythos server provides WebDAV access to content, which is vital for the institutional repository:

- Creating and modifying content in the Xythos server.
In some integration scenarios, providing the mechanism for the institutional repository to access content and its metadata. This mechanism may also be important if you want to move content from the Xythos server into the institutional repository.

Providing an alternate, desktop-based way for content consumers to browse and access material in the institutional repository.

WebDAV access is always available; the important question is how to leverage it:

1. Through operating system-level WebDAV implementations, such as Microsoft’s Web Folders extension to Microsoft Explorer.
2. Through the Xythos Drive, an optional client application that gives Windows users a more familiar desktop experience, while being able to access features of the server (for example, tickets and versioning) from the desktop.
3. In code written to access the Xythos server from the institutional repository, in some custom integration scenarios.

**Categorization**

The document class features of the Xythos Enterprise Document Manager and WebFile Server provide a way to categorize content and apply a particular set of attributes. In other words, document classes are the most direct way to identify content to be published in the institutional repository and enforce the required metadata (author, title, etc.).

**Information lifecycle management**

Content stored in the "records management space" of the Xythos Enterprise Document Manager can have a particular lifecycle defined for it. One step in that lifecycle may be an export to another application, including the metadata that external application requires. In other words, these information lifecycle capabilities provide a natural mechanism for declaring content to be included in the institutional repository, and then providing the metadata needed to properly catalog it.

**E-mail archiving and external content**

“Records” in the “records management space” can include standard electronic documents, such as .DOC and .PDF files, as well as e-mails. Content not stored in the Xythos server, such as physical documents, web sites, and database records, can also be created as records, with the locator information needed to access this content.

These features let you use the Xythos server as a platform for consolidating and managing content from different sources before publication in the institutional repository. For example, library materials on microfiche or other non-electronic formats can be included. So, too, can content that is impossible to translate into digital format, such as lab samples.

**Access control lists**

If you want to enforce security on individual pieces of content, access control lists (ACLs) can restrict the privileges needed to create, modify, access, and manage institutional repository content.

**Tickets**

Tickets provide a secure mechanism for sharing, via a temporary URL, content within the Xythos server. Users do not have to be provisioned in the system to access a file or folder via the ticket. The ticket can expire after a certain period, and it can require a password to access content through it.
With very little effort, you can use tickets as a mechanism for external applications to access content in the Xythos server. The external application—in this case, the institutional repository—might generate a temporary URL, to give someone clicking on a link to a file access to it.

**APIs for customization and integration**

Whatever integration strategy you select, there is more or less likelihood that you will need to access the Java-based APIs of the Xythos WebFile Server. The more sophisticated the integration, the greater the probability that you will use these APIs to customize the behavior of the Xythos server as a source of institutional repository content, integrate the Xythos server with shared infrastructure components (for example, an enterprise portal), or build custom UIs that support the institutional repository (or, in one scenario, become the institutional repository). The APIs can perform a variety of tasks, such as automated publication and custom integration with other applications. The Java APIs are always available, if needed—but depending on the scenario, may not be needed at all.

**LDAP integration**

A critical part of any integration scheme is LDAP support. Authentication, user and group lists, and single sign-on (SSO) are services that both the institutional repository and the Xythos server need to use in tandem. Xythos already provides an out-of-the-box mechanism for LDAP authentication and directory services, and single sign-on is a common, relatively simple customization. (Given the broad array of systems with which Xythos has been integrated, there is no “one size fits all” SSO module that works with all of them.)

**Portal deployment**

Xythos provides three JSR-168 compatible portlets that let you browse, search, download, and upload content. In other words, these provide a portal-based way to contribute content that may end up in the institutional repository, or search its contents (if they stay in the Xythos server).

Many customers and partners write their own portlets, since they want to match the look and feel of their own portal, or they want to encapsulate a particular business process in the portlet. For the institutional repository, you may want to create specialized portlets for content contribution, browsing and searching by metadata, or viewing newly-added content. Writing these kinds of portlets is a relatively simple development project, using the Xythos WebFile Server APIs.

**Import/export**

The 6.0 version of the Enterprise Document Manager, WebFile Server, and Digital Locker added an XML-based import/export framework for moving content and its metadata into and out of the Xythos server. Import/export can play a part in many institutional repository deployments, particularly if the institutional repository has an XML-based mechanism for automatically importing content.

**Scanning integration**

The Xythos server is an ideal back end to a document capture solution. Many desktop scanning tools can save content directly into the Xythos server, via the Xythos Drive or the operating system-level WebDAV. For high-end document capture requirements, including indexing (entering metadata at the point of capture), the Captiva plug-in to the Xythos server provides additional options. Many Ricoh multi-functional devices (MFDs) can use a connector to directly store scanned content in the Xythos server.

 Needless to say, these options simplify one of the common challenges in building an institutional repository, dealing with the large amount of content that is still printed on paper. Even if you do not convert all these documents into digital format right away, or you still need to manage the
original physical copies, you can use the physical document management features in Xythos Enterprise Document Manager 6.0 to include them in your institutional repository.

**Putting it all together**

Since there is more than one approach to making the Xythos server part of an institutional repository, this section describes a few representative scenarios. While none of them represent the “right” (or only) approach, you will probably recognize the one that best fits your needs.

**Scenario #1: Loose integration**

Say that you have an institutional repository, such as DSpace, already in place. How might content from Xythos appear in the institutional repository?

The loose integration maintains the content in Xythos, while using the institutional repository as a front end to that content. The institutional repository contains a hyperlink to the content in Xythos, using either the long URL format (for example, http://www.state.edu/Schools/Biological%20Sciences/Research%20idea.doc) or the shorter Intellilink format (such as http://www.state.edu/xythoswfs/webui/_xy-4328_1).

You have at least two ways of getting the link into the institutional repository. The brute force method involves no customization: users copy the link from the Xythos web UI, and paste them into the institutional repository. If you want to make life easier on these contributors, you can build a simple browsing widget, perhaps using WebDAV to retrieve directory listings, to select a document or folder that you want to link into the institutional repository. It is also possible to build custom code on the Xythos server to push content or identify content to be pulled into the Institutional Repository.

Institutional repository applications may already have the ability to refer to content by URL. If not, they may have underlying mechanisms to virtualize storage. For example, DSpace provides support for the Storage Resource Broker (SRB), an open source storage virtualization framework.

<table>
<thead>
<tr>
<th>Option #1: Loose integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add link to content in the Xythos server</td>
</tr>
<tr>
<td>Xythos server</td>
</tr>
<tr>
<td>Browse and search to locate content stored in Xythos server</td>
</tr>
</tbody>
</table>

**SECURITY**

Obviously, security is important on at least two levels in this arrangement:

- Single sign-on. Users will want to be able to log into the institutional repository and access Xythos-managed content without having to log in again to the Xythos server.
• **Security on the link.** The content must be accessible through the link entered into institutional repository application.

Since it’s nearly impossible to provide a “one size fits all” approach to single sign-on, the exact steps are beyond the scope of this white paper. Suffice it to say, for now, that many customers and partners have implemented single sign-on. For more information, or to ask questions about this topic, go to [http://developer.xythos.com](http://developer.xythos.com).

You can provide security on the link in two ways:

• **Change the ACL settings in Xythos before adding the link in the institutional repository.** This step may be as simple as making the content Public-readable. However, it does require the additional step of changing the ACL, something that users are likely to forget.

• **Provide short-term tickets to the institutional repository application.** Another option is to let the institutional repository application request a short-term ticket, valid for only as long as the user needs to download the content. Northwestern University implemented this approach to provide a loose integration between Xythos Digital Locker and the Blackboard learning management system. Details on this approach are slated to be published on [http://developer.xythos.com](http://developer.xythos.com). Meanwhile, Xythos representatives are happy to discuss how this approach might work.

**METADATA**

What about the metadata? One of the complaints that many users have when using a separate application, such as an enterprise portal, on content stored in a different application, is that the publication layer doesn’t have an automatic way of using the metadata that already exists. For example, users have diligently entered information about the author, title, ISBN number, and publication date of documents stored in the Xythos server, why should they enter the same information all over again in the portal or digital repository?

Currently, Xythos does not have a packaged “connector” to stream metadata from the Xythos server into a particular institutional repository. However, there are some relatively straightforward ways to make this information available:

• **WebDAV.** The WebDAV protocol includes APIs for retrieving properties and their values. These include both standard file system attributes, such as size, and custom attributes, such as **Author** or **Title**. All versions of the Xythos server support WebDAV.

• **The Java API.** The WebFile Server’s Java API (which, by the way, is the same API used to build the Digital Locker and Enterprise Document Manager applications) provides mechanisms for retrieving properties and their values. This could be anything from event handles, UI customizations, or a custom servlet to help push or pull metadata or to provide a custom method to export automated or manually. Using the Java API depends on having a Suite license for the Digital Locker or Enterprise Document Manager versions of the product, or having the WebFile Server platform on its own.

• **Import/export.** The new import/export framework, available in the 6.0 version of the Xythos server, can provide an XML representation of many aspects of content, including metadata. Someone could manually export this information on a regular basis, or create an automated process to create scheduled XML output for content to be published in the institutional repository. Import/export is available in all versions of the Xythos server.

• **RSS.** Although the planned RSS capabilities of the Xythos server are not yet available, adding an RSS feed to Xythos content is a relatively simple customization. (An example is already provided at [http://developer.xythos.com](http://developer.xythos.com).) The institutional repository might read metadata values through an RSS feed, once the institutional repository becomes a “subscriber” to that particular piece of content. Implementing RSS on your own depends on having access to the Xythos APIs, which in turn requires a Suite license for the Digital
Locker or Enterprise Document Manager, or just the WebFile Server. Since the RSS standard allows extensible metadata, converting Xythos metadata into RSS tag information is relatively simple.

Which approach do we recommend? WebDAV has many virtues, including being a well-established standard. Xythos already has several years’ experience supporting customers and partners who use the search component of the WebDAV standard, DASL (DAV Search and Locating), to integrate the Xythos server with other applications. DASL, in turn, includes the search functions needed to extract metadata, and can support other functions of the institutional repository.

If WebDAV does not cover your requirements, the alternative depends largely on your project. For example, if you are already highly familiar with the WebFile Server APIs, they might provide the easiest mechanism for communicating metadata. If the institutional repository already has an XML import mechanism, translating the XML export from a Xythos instance might be the easiest approach.

RECORDS MANAGEMENT
The loose integration approach gives a strong reason to use the records management features of Xythos Enterprise Document Manager 6.0. By declaring a record, you are saying that the contents of a document will not change, and its other attributes, such as its metadata, are also unlikely to be modified. Given the way that the records management features work, the location—a particular folder corresponding to a record category—will change only under very rare circumstances.

Therefore, the links and metadata created in the institutional repository is more likely to remain valid if you move content from the “collaboration space” (the everyday part of the Xythos server) into the “records management space.” Only a very small number of people will have the rights to change anything in the RM space, and only in unusual circumstances. These privileged users may also be the same people who manage the institutional repository, such as the staff of the sponsored research office or the library.

PROS AND CONS
The major advantage of this approach is storage consolidation. Rather than creating a duplicate copy of content already stored and managed in the Xythos server, you keep the single, authoritative copy there. This approach also leaves the people running the institutional repository a great deal of flexibility for linking content in other URL-addressable locations, such as a web site.

The most significant disadvantage depends on how loose the integration is. Without single sign-on, an easy way to browse to a document you want to include in the institutional repository, an automatic way (such as tickets) to handle the security on shared content, or a mechanism for transmitting metadata already stored in the Xythos server, the institutional repository might be hard to use. However, in the first phase of integration, that might be acceptable, as long as you plan to add some of these capabilities in later phases.

The pros and cons depend on what “loose” integration means in practice. The features provided might be quite powerful, as great as in “tight” integration. For example, the ease of publication, and the number of options provided, could equal or excel those in the institutional repository alone.

SUMMARY
The following table summarizes the options for addressing institutional repository requirements, using the “loose integration” approach.
### TABLE 1. LOOSE INTEGRATION

<table>
<thead>
<tr>
<th>REQUIREMENT</th>
<th>OPTIONS</th>
</tr>
</thead>
</table>
| Browse, metadata               | Browse through institutional repository’s UI.  
                             Build a custom “browse by metadata” UI for Xythos.  
                             Use the metadata column headings in the RM space to see metadata values.                                                               |
| Contribution, automated        | Handled through a staging directory that the institutional repository accesses through WebDAV, RSS feed, or other mechanism.                   |
| Contribution, manual           | In the institutional repository, add a link to content stored in Xythos.                                                                   |
| Metadata, basic (author, title, etc.) | Create this metadata in Xythos, applying it to a document class or records.  
                             Optionally, translate this information into institutional repository metadata at the time of submission. |
| Metadata, standard (OAI, etc.) | Create this metadata in Xythos and apply to a document class or records.  
                             Optionally, translate this information into institutional repository metadata at the time of submission. |
| QA after publication           | Depends on what features are available in the institutional repository.                                                                  |
| QA before publication          | Content and metadata could be reviewed in Xythos first.  
                             Alternately, they would be reviewed through the institutional repository application. The reviewer would follow the link to the content in Xythos.  
                             Depending on whether the institutional repository extracts Xythos metadata, the metadata review could be handled on the Xythos or institutional repository side. |
| Search, content                | Institutional repository search UIs include search of content in Xythos.  
                             Alternately, use a federated search tool, such as Google OneBox.                                                                      |
| Search, metadata               | Search the metadata through the institutional repository UIs.                                                                               |
| Security, content-level access control | Primarily controlled through the institutional repository. Sharing settings on Xythos may still be enforced, or access may be allowed through something like an “instant ticket” that the institutional repository generates. |
| Security, single sign-on       | If there nothing like an “instant ticket” feature that gates security through the institutional repository, consumers and contributors will need SSO to access both systems at the same time. |

---

**Scenario #2: Tight integration**

Integration between Xythos and an institutional repository is “tight” if content actually moves from the Xythos server into the institutional repository. This approach might be preferable if an institutional repository already exists, or there is a business reason for keeping content in the institutional repository separate from “live” content.
In this case, the emphasis is on the “feed” from the Xythos server into the institutional repository. In some fashion, users declare that the information is ready for publication in the institutional repository. How the users make this declaration, and what happens next, can happen in a few different ways:

- Manual upload. The users manually upload the content from the Xythos server to the institutional repository.
- Automated upload from a staging area. The Xythos server has some location where users put content to be uploaded into the institutional repository. In this case, you need to leverage the event listener APIs to detect when new content is placed in this staging area. A custom servlet would provide the easiest mechanism for leveraging the APIs in this fashion.
- Mark content for automated upload. Rather than copying or moving content, users set some flag to indicate that the file or folder is ready for publication to the institutional repository. Many options exist here, such as placing content in a particular staging directory, applying a particular document class, or setting a custom property (for example, Ready for publication).
- Transfer or accede. The Enterprise Document Manager 6.0 includes features that are designed, following the DoD 5015.2 records management standard, for export between Xythos and another system.

**Option #2: Tight integration**

Content fed from Xythos into the institutional repository

Xythos server ➔ Institutional repository

Users browse and view imported content from the institutional repository only

**TRANSFER AND ACCESSION**

Since the transfer/accede option is new in the 6.0 version of the Xythos Enterprise Document Manager, it requires some further explanation. Once any content—a PDF file, an e-mail, or a “physical record”—is stored in the records management section of a 6.0 Enterprise Document Manager instance, the content begins a pre-defined lifecycle. One way that each lifecycle phase might end is a temporary transfer to another system, or a permanent export (or accession) to the other system.

In both cases, metadata moves with the content. The metadata appears in an XML file that accompanies any electronic content that will be exported. The XML structure is the same for the import/export mechanisms; the only difference is what sort of XML content gets generated.

If you transfer a record, you get the standard file system metadata, plus the metadata that defines the important information about the record (author, subject, date filed, etc.). Obviously, this information is useful to the institutional repository. Therefore, you might define the metadata for records to include the information that the institutional repository needs.
If you accede the record, you fill out a template that defines a different set of metadata than what is already defined in the Xythos server. This template may represent the metadata that the institutional repository needs.

Some institutional repositories already use the term “accede” to describe how they ingest information from other systems. Therefore, the people who designed systems like DSpace already thought about this process of exporting content from a system like the Xythos Enterprise Document Manager into their repositories. The records management features, therefore, are a natural business and technology fit for institutional repository integrations.

**SECURITY**

The security concerns in this integration model are slightly different from the “loose integration” scenario. Single sign-on (SSO) is still a concern, since you may not want to impose the extra login during the publication process. However, since the content is not linked from the institutional repository to the Xythos server, security on the link is no longer a concern. The user will still need to have read permissions on the content, obviously.

Another major security question arises if you plan on deleting the content from the Xythos server after copying it to the institutional repository. In that case, the user will need to have Delete privileges on the content in Xythos. This additional security check won’t be necessary if, in your organization, it doesn’t matter if a copy of the content remains in the Xythos server.

Depending on the implementation, you may not have the same security restrictions. The WebFile Server APIs provide enough flexibility so that custom server code, such as a servlet, does not have to be restricted to a user’s access control. For example, the Xythos server could treat the institutional repository as a “trusted source,” possibly through a custom authentication method. The Xythos server would not care who the end user is, as long as the security mechanism invoked indicated that it was a valid request from the institutional repository.

This approach could provide many ease of use or manageability benefits, such as automatically eliminating the duplicate copy of a document after its publication in the institutional repository. Alternately, the publication process could modify content in the Xythos (for example, set a flag in Xythos indicating that the content was published in the institutional repository, even if the user did not normally have the privileges to make these modifications.

**METADATA**

The metadata migration options remain the same as in the loose integration scenario. The XML export option gains some importance, if you plan on using the transfer and accede features described above.

**PROS AND CONS**

The major pro of the tight integration is the emphasis it places on the ease of user experience. As a contributor, I don’t have to leave Xythos to publish something in the institutional repository, as long as the “feed” works. As a content or site manager, I don’t have to leave the institutional repository to stay on top of the new content entering the system. Issues like single sign-on must be addressed immediately, instead of making them a Phase II or Phase III project.

The major con to this approach is the cost of maintaining the integration. If there is no packaged “connector,” you’ll have to maintain the code that tightly connects the two systems. Whenever you upgrade one component or the other, you’ll have to revisit the code that you wrote.

If you do not automatically delete content from Xythos any content published in the institutional repository, you may be left with duplicate copies across both systems. This situation may or may not present a problem, depending on your organization. Some might prefer to have the
duplicates, if someone will be updating and re-publishing it at a later date. Others may feel uncomfortable with any stray duplicates.

However, the benefits of the tight integration can often outweigh the costs. The use case determines if the scales swing in the right direction. For example, Wikipedia has shown that the trust model about content contribution can work. If you trust your contributors, you don’t have to build into the integration a quality control step before publishing any content in the institutional repository. Presumably, the contributors know what they’re doing, so you can step back and let the feed work without human intervention. You may want to go back and check on the results, but the tight integration doesn’t have to create a stranglehold on either content contribution or code maintenance.

**SUMMARY**

The following table summarizes the options for addressing institutional repository requirements, using the “tight integration” approach.

<table>
<thead>
<tr>
<th>TABLE 2. TIGHT INTEGRATION</th>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REQUIREMENT</strong></td>
<td><strong>OPTIONS</strong></td>
</tr>
<tr>
<td>Browse, metadata</td>
<td>Browse through the institutional repository UI.</td>
</tr>
<tr>
<td>Contribution, automated</td>
<td>Use WebDAV to import content staged in Xythos. Use custom solution with Xythos APIs to push or allow pulling of content into the institutional repository (events, custom servlet). Use the XML import features of the institutional repository to “harvest” content that Xythos has exported with XML metadata information.</td>
</tr>
<tr>
<td>Contribution, manual</td>
<td>Provide a browse widget for selecting content in the Xythos server. Through a custom Xythos UI, push into the institutional repository. Through the desktop or the standard Xythos web UI, move into a staging area from which the institutional repository will “harvest” new content. Use WebDAV or another mechanism for importing the content, plus any metadata already defined for it.</td>
</tr>
<tr>
<td>Metadata, basic (author, title, etc.)</td>
<td>Create document classes in the non-RM space in Xythos that contain this information. Add this information to the definition of records in the RM space, or add it to the accession template.</td>
</tr>
<tr>
<td>Metadata, standard (OAI, etc.)</td>
<td>Create document classes in the non-RM space in Xythos that contain this information (Enterprise Document Manager only). Use custom properties (name/value pairs), if you are using the Digital Locker version of the server. Add this information to the definition of records in the RM space, or add it to the accession template.</td>
</tr>
<tr>
<td>QA after publication</td>
<td>Use existing QA features in the institutional repository to review imported content before publication.</td>
</tr>
<tr>
<td>QA before publication</td>
<td>Enforce validation rules on any automatically-imported content. Enforce similar rules when manually importing content from the institutional repository UI. Workflows in Xythos, when approve, publish content</td>
</tr>
</tbody>
</table>
Scenario #3: One system

Many organizations don’t want to add a separate application every time someone has a new use for content that is stored in the Xythos server. A separate institutional repository application is yet another system for IT to deploy and maintain. Users need to have access to this application, and then learn it.

Faced with these requirements, some organizations decide that building a “digital library” view on the Xythos server is a better approach. TimeWarner Retail Systems, Inc., for example, built just such an application, an institutional repository for a media company, on the Xythos WebFile Server.

What then is the important functionality of an institutional repository? Here they are, with their importance (showstopper, must have, or nice to have) noted.

- A UI for browsing by metadata. For example, the UI needs to let users drill into content by author and subject. (Showstopper)
- Search by metadata. Users need to be able to search for content by the assigned metadata. Some attributes may allow freeform entry (for example, title), while others might be constrained to pick lists (such as MIME type). (Showstopper)
- Contribution (manual). Getting content into the institutional repository can be a manual process, in which users add new content piece by piece. (Showstopper)
- Contribution (automated): Some institutional repositories also have “harvesting” features that can import content from other systems in bulk. (Must have)
- QA before publication: Some organizations may want to have the people responsible for the institutional repository perform QA on any contributions before they become available. (Nice to have)

Content search is a definite plus, especially since it does not appear in most institutional repositories. Any institutional repository built on Xythos, therefore, could provide immediate value by leveraging the content search features.
PROS AND CONS

The major downside of this approach is writing the institutional repository functionality from scratch. Fortunately, institutional repositories are relatively simple systems, compared to other applications (for example, a CRM system) that might use the content stored in the Xythos server. The institutional repository is a particular metadata-driven view of content, designed for people who will be browsing and searching for particular reference material.

The major upside of this approach is simplification for IT and users alike. The IT department will have to maintain the custom code, but it can re-use the administration and development skills needed when the Xythos server appears in other projects, such as secure collaboration. Users won't have to be provisioned on another application, and the amount of extra functionality they will need to learn on top of the Digital Locker and Enterprise Document Manager depends on how fancy you get with the institutional repository functionality.

The psychological value of telling users, “This is the place to go for all important documents,” is quite high. Information silos remain one of the biggest challenges facing IT departments today. Building the institutional repository on the Xythos server eliminates one of these silos.

SUMMARY

The following table summarizes the options for addressing institutional repository requirements, using the “one system” approach.

<table>
<thead>
<tr>
<th>TABLE 3. ONE SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQUIREMENT</td>
</tr>
<tr>
<td>Browse, metadata</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Contribution,</td>
</tr>
<tr>
<td>automated</td>
</tr>
<tr>
<td>Contribution,</td>
</tr>
<tr>
<td>manual</td>
</tr>
<tr>
<td>Metadata, basic</td>
</tr>
<tr>
<td>(author, title, etc.)</td>
</tr>
<tr>
<td>Metadata, standard</td>
</tr>
<tr>
<td>(OAI, etc.)</td>
</tr>
<tr>
<td>QA after publication</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Future directions

Xythos is planning new features for the different Xythos server products, the Xythos Drive, and other products that may figure into your institutional repository plans. Here are a few examples:

- “Browse by metadata” pages in the Digital Locker and Enterprise Document Manager web UIs.
- Enter metadata when uploading content through the Xythos Drive.
- RSS feeds for the contents of folders.
- Workflow approvals drive publication in a metadata-driven view of the Xythos server.

If you are curious about these future plans, contact your Xythos representative, or post a question at [http://developer.xythos.com](http://developer.xythos.com). We mention these ideas not to complicate your thinking about adding the Xythos server to your institutional repository, but to show you the value of your investment in Xythos technology. Over time, the number of tools that may be used to address institutional repository requirements will only expand.

### Conclusion

As you can tell, Xythos provides technology that fits snugly into many institutional repository scenarios. The Xythos server can be the starting point for content that is published in the institutional repository. Whether this content moves into the institutional repository, or stays in the Xythos server, is your choice. Given the stark simplicity of many institutional repositories, it may even be worth thinking about building the institutional repository as an alternate or supplemental application interface on an existing Xythos instance.

Whatever your preferred approach, Xythos is ready to help. Our mission as a company is to provide technology that people will use, for institutional repositories as well as other projects.
Appendices

Appendix A: Simple DSpace example

Using the import/export framework, you can add the metadata that an institutional repository requires to the Xythos server. Here is some sample XML, which creates a document class that includes several attributes that the DSpace application uses. (Click here for the complete DSpace metadata model.)

If you want to use this example, follow these steps:

1. Copy this information into a new XML file.
2. Use the Xythos administrator UI to import this information.
   If you don’t have access to a Xythos instance, ask your Xythos representative about the evaluation version of Xythos Enterprise Document Manager.
3. Apply this document class to a folder.
   Be sure to make this document class the default for the directory.
4. Upload a new file into the folder through the Xythos web UI.
   You’ll now be asked to add this metadata before uploading the file.

The same metadata may also be applied to the definition of a record in Xythos Enterprise Document Manager, or the accession template for content exported to an institutional repository.

```xml
<?xml version="1.0"?>
<DocumentRepository>
  <Property name="dc.contributor.author">
    <Description><![CDATA[IR example: Author(s)]]]></Description>
    <Datatype><![CDATA[short-string-array]]></Datatype>
    <ClassificationNamespace/>
    <isCaseSensitive>false</isCaseSensitive>
    <isReadableOutsideAPI>true</isReadableOutsideAPI>
    <isVersioned>true</isVersioned>
    <isWriteableOutsideAPI>true</isWriteableOutsideAPI>
    <isProtected>false</isProtected>
  </Property>
  <Property name="dc.identifier.uri">
    <Description><![CDATA[IR example: URI]]></Description>
    <Datatype><![CDATA[short-string]]></Datatype>
    <ClassificationNamespace/>
    <isCaseSensitive>false</isCaseSensitive>
    <isReadableOutsideAPI>true</isReadableOutsideAPI>
    <isVersioned>true</isVersioned>
    <isWriteableOutsideAPI>true</isWriteableOutsideAPI>
    <isProtected>false</isProtected>
  </Property>
  <Property name="dc.title">
    <Description><![CDATA[IR example: Title]]></Description>
    <Datatype><![CDATA[short-string]]></Datatype>
    <ClassificationNamespace/>
    <isCaseSensitive>false</isCaseSensitive>
    <isReadableOutsideAPI>true</isReadableOutsideAPI>
    <isVersioned>true</isVersioned>
    <isWriteableOutsideAPI>true</isWriteableOutsideAPI>
    <isProtected>false</isProtected>
  </Property>
  <Property name="dc.description.abstract">
    <Description><![CDATA[IR example: Abstract]]></Description>
    <Datatype><![CDATA[string]]></Datatype>
    <ClassificationNamespace/>
    <isCaseSensitive>false</isCaseSensitive>
    <isReadableOutsideAPI>true</isReadableOutsideAPI>
    <isVersioned>true</isVersioned>
    <isWriteableOutsideAPI>true</isWriteableOutsideAPI>
  </Property>
</DocumentRepository>
```
Building an institutional repository with Xythos

Copyright © 2008 Xythos Software

<isProtected>false</isProtected>

</Property>

<Property name="dc.subject.ddc">
  <Description><![CDATA[IR example: Dewey Decimal Number]]></Description>
  <Datatype><![CDATA[short-string]]></Datatype>
  <ClassificationNamespace/>
  <isCaseSensitive>false</isCaseSensitive>
  <isReadableOutsideAPI>true</isReadableOutsideAPI>
  <isVersioned>true</isVersioned>
  <isWriteableOutsideAPI>true</isWriteableOutsideAPI>
  <isProtected>false</isProtected>
</Property>

<Property name="dc.date.copyright">
  <Description><![CDATA[IR example: Copyright date]]></Description>
  <Datatype><![CDATA[timestamp]]></Datatype>
  <ClassificationNamespace/>
  <isCaseSensitive>false</isCaseSensitive>
  <isReadableOutsideAPI>true</isReadableOutsideAPI>
  <isVersioned>true</isVersioned>
  <isWriteableOutsideAPI>true</isWriteableOutsideAPI>
  <isProtected>false</isProtected>
</Property>

<Property name="dc.format.mimetype">
  <Description><![CDATA[IR example: Mimetype]]></Description>
  <Datatype><![CDATA[short-string]]></Datatype>
  <ClassificationNamespace/>
  <isCaseSensitive>false</isCaseSensitive>
  <isReadableOutsideAPI>true</isReadableOutsideAPI>
  <isVersioned>true</isVersioned>
  <isWriteableOutsideAPI>true</isWriteableOutsideAPI>
  <isProtected>false</isProtected>
</Property>

<DocumentClass name="Institutional repository">
  <Description><![CDATA[Metadata for an institutional repository. Based on the DSpace metadata scheme. Not all attributes are replicated from DSpace in this example.]]></Description>
  <isProtected>false</isProtected>
  <Properties>
    <Property name="dc.contributor.author">
      <isRequired>false</isRequired>
      <isDeletable>true</isDeletable>
    </Property>
    <Property name="dc.title">
      <isRequired>false</isRequired>
      <isDeletable>true</isDeletable>
    </Property>
    <Property name="dc.identifier.uri">
      <isRequired>false</isRequired>
      <isDeletable>true</isDeletable>
    </Property>
    <Property name="dc.description.abstract">
      <isRequired>false</isRequired>
      <isDeletable>true</isDeletable>
    </Property>
    <Property name="dc.date.copyright">
      <isRequired>false</isRequired>
      <isDeletable>true</isDeletable>
    </Property>
    <Property name="dc.format.mimetype">
      <isRequired>false</isRequired>
      <isDeletable>true</isDeletable>
    </Property>
  </Properties>
</DocumentClass>
</DocumentRepository>
**Appendix B: Complete DSpace example**

The XML content for the complete implementation of the DSpace metadata model is too long to be included here. To download an XML file containing the complete DSpace metadata, [click here](http://developer.xythos.com). If this link does not work, visit [http://developer.xythos.com](http://developer.xythos.com) for more details.

For a full list of the attributes in the DSpace metadata model, [click here](http://developer.xythos.com).