HyperContent v1.4 manual

An exhaustive reference manual for HyperContent Version 1.4, including chapters dedicated to installation, customization, use and technology references.

by

Alex Vigdor

published by

Columbia University Administrative Information Services

last updated Sep 15, 2005 5:09:39 PM EDT

HyperContent 1.4

HyperContent is a robust content management framework fronted by an extensible set of uPortal channels and backed up by a full-featured repository and multithreaded publishing engine. HyperContent is a complete solution for the rapid development of completely custom, well structured web sites, requiring knowledge of XML, DTDs and XSL - no Java programming required.

Following the release of HyperContent v1.0, which implemented this new framework for managing web sites from within a uPortal channel, and v1.1, which extended the framework with more robust editing capabilities, metadata support and enhanced performance, v1.2 featured a major overhaul of the user interface with many powerful new features, including

- a WYSIWYG editor for free-form text
- Full content and metadata searching
- New contextual functions for creating, copying and moving documents and folders
- VCARD contact information editing, with configurable LDAP feed

Site developers found new features geared towards simplifying their work:

- maintain DTDs and XSL imports right in the repository
- upload and preview XSL in two clicks
- publication metrics identify possible site implementation bottlenecks
- new editor framework simplifies the development of custom editing UIs

HyperContent version 1.3 added multi-lingual spell checking ability (with built-in dictionaries for American and British English, French and German), in-browser photo cropping and resizing, image thumbnails, SFTP publishing support, and more. Version 1.4 adds a new site navigation management system.

HyperContent is widely used at Columbia University to manage departmental web sites; this release has been tested, tweaked and proven in a heavily used, multi-project environment. Don't hesitate to take it for a test drive to meet your own web site management needs!

Table of Contents

Overview 1

Features 2
Architecture 4
Process 5
Release Notes 8
License 14

Getting Started 15

System Requirements 17
Installation 18

Using the Content Manager 19

Top Navigation 22
Search 23
Directory tree navigation 25
Details screen 26
Create a new file 28
Copying & Moving files 30
Admin section 31
Output section 33
References 34
Revisions section 35
Report screen 36
Permissions 39

Authoring Content 40

Dublin Core Metadata 42
WYSIWYG & XML 43
Upload & Download 44
Plain Text 45
VCard (Contact Info) 46
Spell Check 48
Images 50
Build Options 52
Navigation / Site Map 53

Developing Sites 54

Project Definition Files 55
Filesystems 56
Path patterns 58
Content Types 61
XML Content 62
RDF: Extensible Metadata 67
Dublin Core Metadata 72
VCard Metadata 77
XML Includes 83
XSL Transformations 86
Filters 88
Non-XML Content 90
Editors 92
Ignored Directories 94
Permissions 95
Navigation and Site Maps 96
Optimizations 100

Glossary 101

Bibliography 103

Overview

The documentation for HyperContent is divided into five main sections; this first section, the Overview, is intended to provide an introduction to HyperContent, including its features, architecture, licensing, and the process of developing a web site. This is high-level information that may provide useful to individuals, educational institutions or companies interested in understanding or evaluating the capabilities of HyperContent, as well as users of previous versions interested in seeing what this new release has to offer. Each section of the documentation is comprised of chapters on specific topics; the other four sections are

Getting Started

This section describes the process of downloading and installing HyperContent. It is intended for a technical audience.

Using the Content Manager

This is documentation for project administrators and content authors describing the administrative functionality of HyperContent.

Authoring Content

This is documentation specifically for content authors, describing the data-entry screens in HyperContent. These screens are accessed from the Content Manager described in the prvious section.

Developing Sites

This section is an complete development and configuration guide for creating complete custom web sites. It is intended for a technical audience.

Features

HyperContent is thorough solution for web content management. The bundled administrative and authoring applications run as channels in a uPortal environment, and are accessible from standard web browsers.

Authoring

HyperContent features a configureable content authoring environment, with built in support for

- WYSIWYG markup (in IE or Mozilla)
- DTD driven XML
- Spell checking (American or British English, French or German)
- Image cropping and resizing
- Dublin Core metadata
- Upload/Download from the Desktop
- VCard (contact info)
- Plain text
- Custom JAVA editors, which can be developed using the provided IEditor framework or standard uPortal IServant implementations

Storage

HyperContent is built on a robust, reusable Filesystem package that includes

- basic file and directory I/O
- user-specific temp files
- saving old file editions
- file and directory locking
- RDF metadata for every file edition and directory
- standard "flat" filesystem support
- FTP and SFTP access for remote filesystems
- read-only access to ZIP files
- search indexing
- API to allow extensibility to other back ends, such as RDBMS or WebDAV

Management

A full set of management functionality is provided by the Content Manager channel:

- repository search
- build, publish and preview
- create, edit, copy, move, compact or delete files

- upload or download zip of any file or directory
- manage authorizations
- view relationships between files
- edit or revert to old file editions

Customization

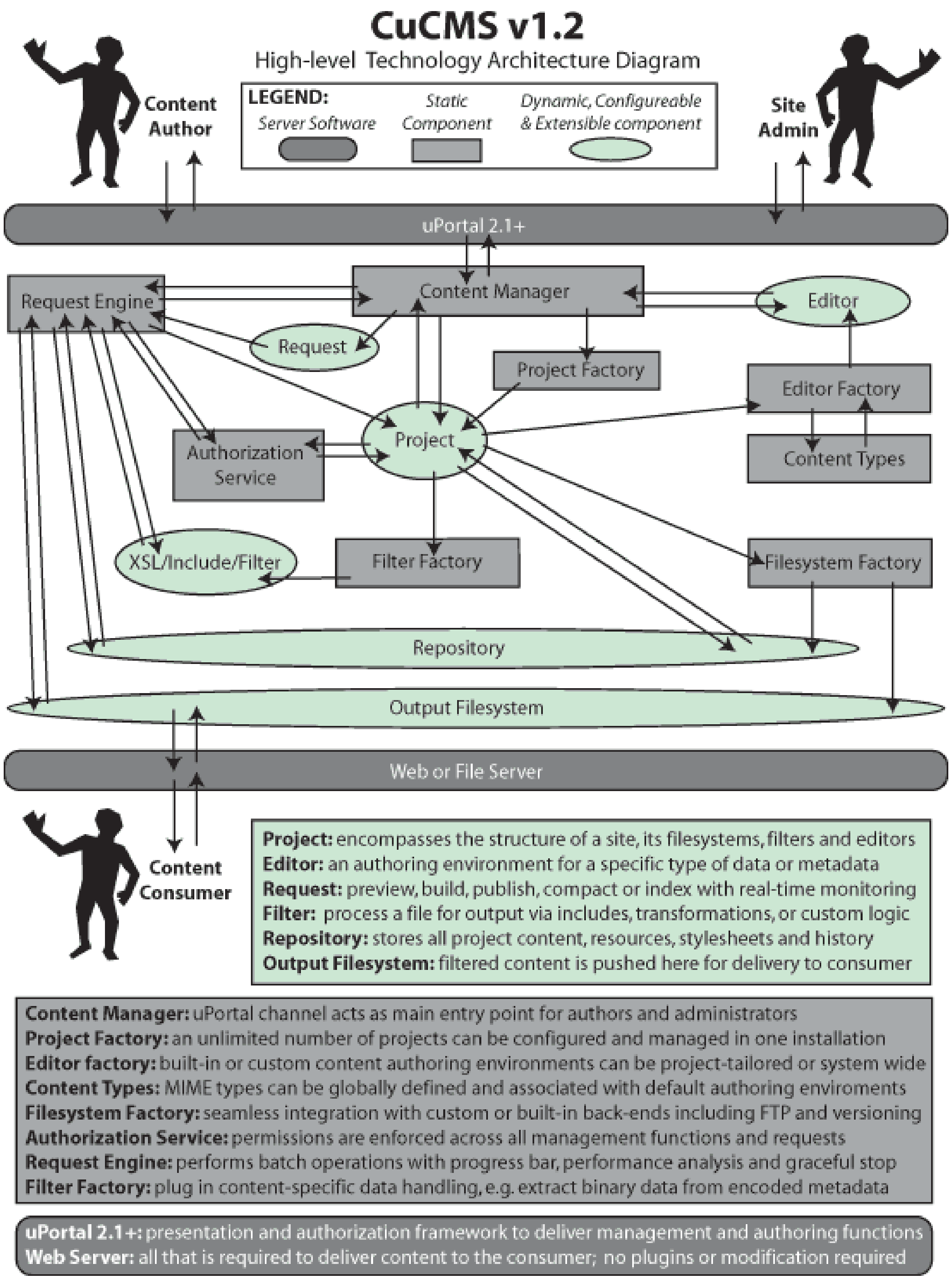
HyperContent provides a multitude of ways to create competely customized web sites and project-tailored authoring environments:

- Project definition grammar to define custom, well-structured sites
- DTD driven authoring for custom XML grammar support
- XSL allows separation of presentation from content
- XML includes allow re-use of data across the site and in different formats
- configureable data and metadata authoring environments
- associate new or known MIME content types with specific editing tools
- JAVA filter framework to create new forms of output
- IEditor framework for project-specific authoring environments
- Filesystem framework allows for custom back-ends

Internationalization

The standard components of HyperContent are completely localizable and UTF-8 transparent, providing full support for use with many different languages. In addition, the IEditor framework has built-in tools to simplify the development of localizable authoring environments.

Architecture



Project: encompasses the structure of a site, its filesystems, filters and editors
Editor: an authoring environment for a specific type of data or metadata
Request: preview, build, publish, compact or index with real-time monitoring
Filter: process a file for output via includes, transformations, or custom logic
Repository: stores all project content, resources, stylesheets and history
Output Filesystem: filtered content is pushed here for delivery to consumer

Content Manager: uPortal channel acts as main entry point for authors and administrators
Project Factory: an unlimited number of projects can be configured and managed in one installation
Editor factory: built-in or custom content authoring environments can be project-tailored or system wide
Content Types: MIME types can be globally defined and associated with default authoring enviroments
Filesystem Factory: seamless integration with custom or built-in back-ends including FTP and versioning
Authorization Service: permissions are enforced across all management functions and requests
Request Engine: performs batch operations with progress bar, performance analysis and graceful stop
Filter Factory: plug in content-specific data handling, e.g. extract binary data from encoded metadata

uPortal 2.1+: presentation and authorization framework to deliver management and authoring functions
Web Server: all that is required to deliver content to the consumer; no plugins or modification required

Process

This chapter is intended to provide a brief, prosaic description of the process of developing a web site with HyperContent.

Information Architecture

The first and most critical stage of developing a web site is determining the information architecture. Things that must be established include what information will be conveyed, what kinds of pages are necessary to present that information, how pages relating to different kinds of information will be grouped, how broad and deep these groupings should be, and how navigation will be performed among the different groupings and page types. These decisions need to be made with a goal of clarity and comprehensibility to the intended audience.

There are many resources to help develop a coherent strategy for information architecture, including an O'Reilly book and a webmonkey tutorial.

Graphical User Interface Design

GUI Design generally starts with pen, paper, and the intention to express an information architecture, from its structure and navigation to the display of specific types of information. The process is iterative, with each set of sketches being a refinement of the last, based on review, feedback and fresh ideas. The end result of the GUI Design process is generally a set of paper, graphical or HTML mockups that exemplify the overall site branding and the presentation of every type of page and navigational component specified by the information architecture. This design ultimately will be distilled down to CSS, image files and HTML gestures. The CSS and image files can be imported directly into HyperContent for site implementation; the HTML gestures will be coded into XSL templates designed to present the structured data stored in the site repository.

Site Implementation

Site implementation requires an information architecture, but may begin without a finished GUI design; as the goal of site implementation is to maintain a separation between content and presentation, much of the work can be done before the presentation is fully developed.

An information architecture is expressed at two levels in HyperContent; at the lowest level, XML DTDs are used to express the structure of data fields. At a higher level a Project Definition document describes file and directory structures, the association of particular XML root elements with specific XML file types, the includes, XSL and filters associated with XML files, support for non-XML content types, and any customizations of the content authoring environment. Project Definitions also describe where the data repository will be, as well as where the site output should go. Project Definitions are described in depth in the Developing Sites section of the HyperContent documentation.

When the site implementation is complete, the end result will be a Project Definition file, a set of XML DTDs and XSL, and any number of image, CSS and other support files which are used to construct the GUI. All of these files can be managed using the Content Manager channel, which is the administrative and authoring tool provided by HyperContent.

Content Development

Once a site is implemented, the Content Manager channel gives content authors everything they need to create, update and manage content in the site. This includes the ability to browse, search, create, copy, move, edit, delete, preview, build and publish files and folders in the repository. All of these activities are subject to authorization, so that administrative users can specify exactly who has what privileges.

During authoring, there is a repetitive cycle of making changes and previewing the result. It is possible to give content authors permission to build content, which runs their changes through the publishing engine and onto the staging server, but not publish it, allowing review by an editor before the content is made publicly available.

The Content Manager channel is described in depth later in this manual, as are the specific authoring tools that ship with HyperContent. HyperContent also provides a complete framework for plugging in custom editors.

Adding on

As time goes on, many web sites need to add additional types of information in order to meet new needs or cater to increasingly sophisticated users. At any time, the project DTDs can be modified to add additional fields or attributes that can be used in existing as well as new XML content. You can easily extend either the breadth or depth of the site with new page types that will live alongside or beneath current data, and it is possible to implement a new look and feel, navigational scheme, or audience-specific output either in place of or alongside the existing site, by modifying or writing new XSL templates.

Release Notes

v1.4 (6/18/2004)

NEW FEATURES:

- Site Navigation management system including XML templating language, navigation generation filter and navigation editor with drag-and-drop reordering.
- Supports uPortal 2.3
- Allow metadata RDF/XML loading from XSLT document() function with special '.rdf' extension on regular file path
- Support for setting group for files pushed to SFTP filesystems
- Support for build-time filters which are run as pre-processing, enabling filtered data to be included by other files.

BUG FIXES:

- dependencies in specific configurations may have been missed causing content to become stale
- WYSIWYG may have not displayed properly under Internet Explorer at times
- if current edition of a file is deleted, most recent remaining edition now becomes current
- update HyperContent DTD url in project definitions
- files downloaded from file details screen are now properly downloaded with filename

v1.3 (3/15/2004)

NEW FEATURES:

- Spell-checker with built in dictionaries for American and British English, French and German and support for custom project dictionaries
- Image Editor allows in-browser cropping and resizing of photos and images, with automatic conversion between JPG, GIF and PNG formats
- Thumbnails automatically generated and displayed for images and files with VCARD photos
- SFTP support for secure publishing
- support uPortal 2.2 dynamic locale selection

BUG FIXES:

- Dates of files in an FTP filesystem may have been inaccurate
- XML editor could fail to update content with Xerces 2.2.1
- jta.jar bundled to workaround uPortal 2.2 installation bug

CHANGES:

- Lots of refactoring and package changes
- increase buffer size and reduce memory use by reusing buffers
- now configured for uPortal 2.2 quick-start installation
- update sample project to demonstrate new features
- update FTP library to 1.2.3
- update WYSIWYG from latest HTMLArea code
- XML includes optimized for speed and memory efficiency by reducing in-memory data copying, and caching and filtering SAX events
- no longer delete output file if build or publish fails
- tweak XML editor GUI for better skin compatibility
- store temp files asynchronously
- enhance performance with custom UTF-8 reader, writer and properties loader
- renamed project from CuCMS to HyperContent!

v1.2 (12/2003)

NEW FEATURES:

- WYSIWYG markup editing with automatic HTML cleanup
- Full content and metadata searching
- VCARD metadata editor with LDAP integration and photo support
- Copy and move files and directories
- Contextual document creation
- Download or upload ZIP of any directory or file in the repository
- Compact command deletes all historical file editions, leaving only current content
- Prune command deletes empty directories from the repository
- Preview, Build or Publish from any node in the repository filesystem
- Force Build and Force Publish refreshes all content
- All requests can now be gracefully stopped
- Performance metrics for Building helps identify site implementation bottlenecks
- DTDs and XSL imports can now be stored right in the repository
- Preview XSL without saving
- Locate repository files by URL of output
- View file dependencies and references
- Ant build file simplifies installation
- Full localization capabilities for presenting GUI in other languages
- Support for non-editioning or prompt-required repositories
- Support for directory metadata, including label
- Configure "ignored" directories to allow nested filesystems
- Caching and performance enhancements
- Custom editor framework simplifies development of authoring environments
- Filter framework allows completely custom data manipulations

BUG FIXES:

- improve stream handling (close streams in finally statement)
- improve FTP filesystem compatibility with solaris
- fix bugs in filesystem copy and move functions that lost some metadata
- double-check permissions before performing commands, eliminate possible exploits
- enable use of Content Manager channel while request is running in background window
- improved filesystem change handling on screen refresh
- fix bug in XML editor where inserts may have occurred in the wrong places
- fix XML editor bug where parsed entities acquired from a system identity would be lost
- reject file or directory names with odd characters
- Content Manager GUI now enables browsers to properly display hand/link pointer over build, publish and preview links

CHANGES:

- new MIT-based license
- Filesystem API: added methods to store search index, copy directory
- upgrade to Jena 2.0
- refactored all editors to use new IEditor and localization frameworks
- permissions policy supports wildcards for new, copy and move functions
- changed ContentTypes.xml format to support new editor framework
- new request threading code improves multiple simultaneous user experience
- use of threadlocal variables in some classes to increase performance
- CMSException can use formatted messages for improved readability and localizability
- improved reporting of errors and warnings during build
- improved pattern recognition allows shallower, more readable directory paths and more strongly enforced structures
- defer filesystem access to support runtime re-configuration
- new icons: forward/back in edit mode, preview, edit and delete, content types, details
- one-click save or discard
- overhaul of file creation GUI; now contextual and more intuitive
- add new GUI sections for create, admin, copy/move, references, output, revisions
- directory GUI widgets are now anchored for improved navigation
- show dublin core title in details screen
- root directory is now visible and selectable in filesystem tree
- temp files now kept in memory during editing to speed up editor-switching
- XML Editor: supports ID attributes, HTML parsing, new process icon, expand/collapse and lock/unlock elements
- much refactoring in publication engine
- removed converting filesystem impl; 1.0 installations must upgrade to 1.1 before 1.2

v1.1 (04/22/2003)

NEW FEATURES:

METADATA SUPPORT:

- New filesystem package is integrated with HP's Jena for file-level RDF metadata support
- New Dublin Core metadata editor works with any file type
- All metadata, including Dublin Core, is accessible for page rendering via XML Includes
- Multiple Editor support allows editing of disparate sets of metadata for the same file

NEW FILESYSTEM API AND IMPLEMENTATION:

- Filesystem-level RDF metadata support
- enhanced for performance
- no uPortal or HyperContent dependencies, so filesystem package can be re-used separately

OASIS XML CATALOG SUPPORT:

- Used to specify rewrite rules for locating DTDs or XML files
- Can be used to support local/offline hosting of documents normally retrieved from a URL
- Complete integration: any DTD, include file, stylesheet or URL referenced from an XSLT document() call can be redirected using the catalog

MULTIPLE EDITOR SUPPORT:

- Specify any number of different data and metadata editors per file
- Editors can be specified at global, project and file levels
- If a single editor fails to initialize, others are still available

ENHANCED BUILD FLEXIBILITY:

- Specify includes on a per-file basis
- Specify multiple forms of output for a page in the same markup
- Replace one file's contents with another at build time
- Retrieve includes, replacements and stylesheets from URLs

HyperContent is now fully UTF-8 transparent for working with worldwide scripts

BUG FIXES:

- File upload properly recognizes equivalent content types

- Files with custom specified editor factories whose content type had no universally specified factory would appear uneditable in Content Manager channel
- DTD parser for XML editor fixed to maintain attribute order and properly handle certain configurations of nested element lists

CHANGES:

- with-baseurl XSL parameter now provides a relative rather than absolute path
- with-property XSL parameter removed: RDF metadata is now available via XML includes
- with-filecomment XSL parameter removed: RDF metadata is now available via XML includes
- with-filename XSL parameter removed: XSL is now guaranteed the presence of cms:source and cms:include tags containing this information, as well as path and content type.
- XML Include elements XmlIncludeWrapper, XmlSource and IncludedFile have been renamed respectively to wrapper, source and include under the namespace "http://www.ais.columbia.edu/sws/xmlns/cucms#"
- The new editioning Filesystem implementation will not work with 1.0 repositories. For backwards compatibility, change the "local-editioning" filesystem definition in /properties/FileSystemConfig.xml to point to the class "edu.columbia.ais.portal.cms.filesystem.ConvertingFileSystemImpl". Each filesystem will be converted on the first attempt to access it. This might take some time. Once all filesystems are converted and verified, you can safely change the implementation back to "edu.columbia.ais.filesystem.impl.LocalEditioningFileSystemImpl"

v1.0.1 (01/03/2003)

BUG FIX:

- "Deny" type permissions were not enforced

v1.0 (12/18/2002)

Initial release of HyperContent.

Features:

- Store, edit and publish files of any type
- Maintain history of file changes
- Recognize relationships between files
- Enforce access controls to projects and files
- Use uPortal as the platform for management functions
- XML based content storage for adaptability, reuse, speed of development
- XSL Transformations to allow multiple output forms of XML content

- XML Includes to minimize data redundancy
- Extensible, customizable framework for file editors
- Publish files to remote servers via FTP

License

Getting Started

HyperContent requires JDK 1.4+ and uPortal 2.1+ to run; even if you do not already have a uPortal installation, you can quickly get up and running with both uPortal and HyperContent by downloading the uPortal 2.3.2 quick start and the HyperContent v1.4 distribution. The HyperContent distribution is pre-configured for installation into the uPortal 2.3.2 quick-start: once both packages are downloaded and unzipped, just drag the HyperContent distribution folder into the uPortal quick-start folder. First, open a command prompt, and run:

```
./ant.sh hsql
```

or on windows:

```
ant hsql
```

Next, in a different command window, execute the following command:

```
./ant.sh -buildfile HyperContent1_4/build.xml
```

or on windows

```
ant -buildfile HyperContent1_4\build.xml
```

This runs the default HyperContent install target, which will deploy media, properties, libraries, repositories and publish the Content Manager channel.

If this completes without any errors, you can start up the portal application server. Before doing this the first time, it is recommended that you increase tomcat's memory allocation, as the default of 64 MB is insufficient for advanced use of HyperContent (such as spell checking and building large sites). You can do this by adding this tag in the uPortal ant script 'build.xml', in the 'tomcatstart' target after the classpath element:

```
< jvmarg value="-Xmx256M"/>
```

You start the application server with the command

```
./ant.sh tomcatstart
```

or on windows

```
ant tomcatstart
```

Access the portal by going to **http://localhost:8080/uPortal** in a browser window. Log in with the username admin and password admin, and go to the Preferences section to subscribe to the Content Manager channel. At this point, you are ready to proceed to the next section of the manual.

If you would like to install HyperContent into an existing installation of uPortal, or want information on other available ant targets, read the Installation chapter for details on how to configure the ant scripts for deployment. If you are upgrading from HyperContent 1.2 or 1.3, it is recommended that you run the "undeploy" ant target from the previous version before upgrading, to eliminate the possibility of library conflicts.

System Requirements

HyperContent is a server-side java application, and is accessed via a browser. The GUI runs as a channel inside uPortal. Therefore, the software requirements on the server side are

- Java version 1.4+
- Web application server (e.g. tomcat)
- Web server (e.g. apache, or tomcat standalone)
- uPortal 2.1.x, 2.2.x or 2.3.x

On the client side, basic functionality is available to any javascript-enabled browser, but WYSIWYG editing requires either

- Internet Explorer 5.5+ (on the PC - 6 recommended)
- Mozilla 1.3+ (Mac, PC, Linux, etc. - 1.5 recommended)

Server hardware
It is highly recommended that you have at least 192 MB of physical memory free and allocated to the JVM running uPortal and HyperContent. HyperContent build performance is constrained equally by disk and CPU speed; while there are no minimum requirements for CPU or disk speed, optimum performance will be achieved with a 1+Ghz processor and a 7200+ rpm drive.

Installation

Installation of HyperContent is automated through the use of an Ant script. The file *build.properties* in the hypercontent distribution directory has three configureable properties which you can use to control where the files are installed:

- ***deploy*** This property should point to the filesystem path of the deployed uPortal web application; this will be used to determine where media and libraries should be installed, as well as allowing HyperContent to reference uPortal libraries for compilation.
- ***common.lib*** This property should point to the common/lib directory of tomcat, or a similar directory that stores libraries available to the uPortal process but not stored under the uPortal web application.
- ***runtime.dir*** This property is misleadingly named on purpose; it is used to specify where the cucms-repositories directory will be installed. It is easiest to set this to the path of the directory which is the context of the application server; this allows HyperContent to use relative paths to reference repositories. If you set this to another directory, you will also need to change the filesystem declarations in the Projects.xml property file, and in the sample project configuration, to point to the appropriate path. The sample project configuration file can be edited with HyperContent running, by opening the HyperContent Project Definitions project and editing the hypercontent-site.xml document under the projects directory.

The file *build.xml* contains the following ant targets:

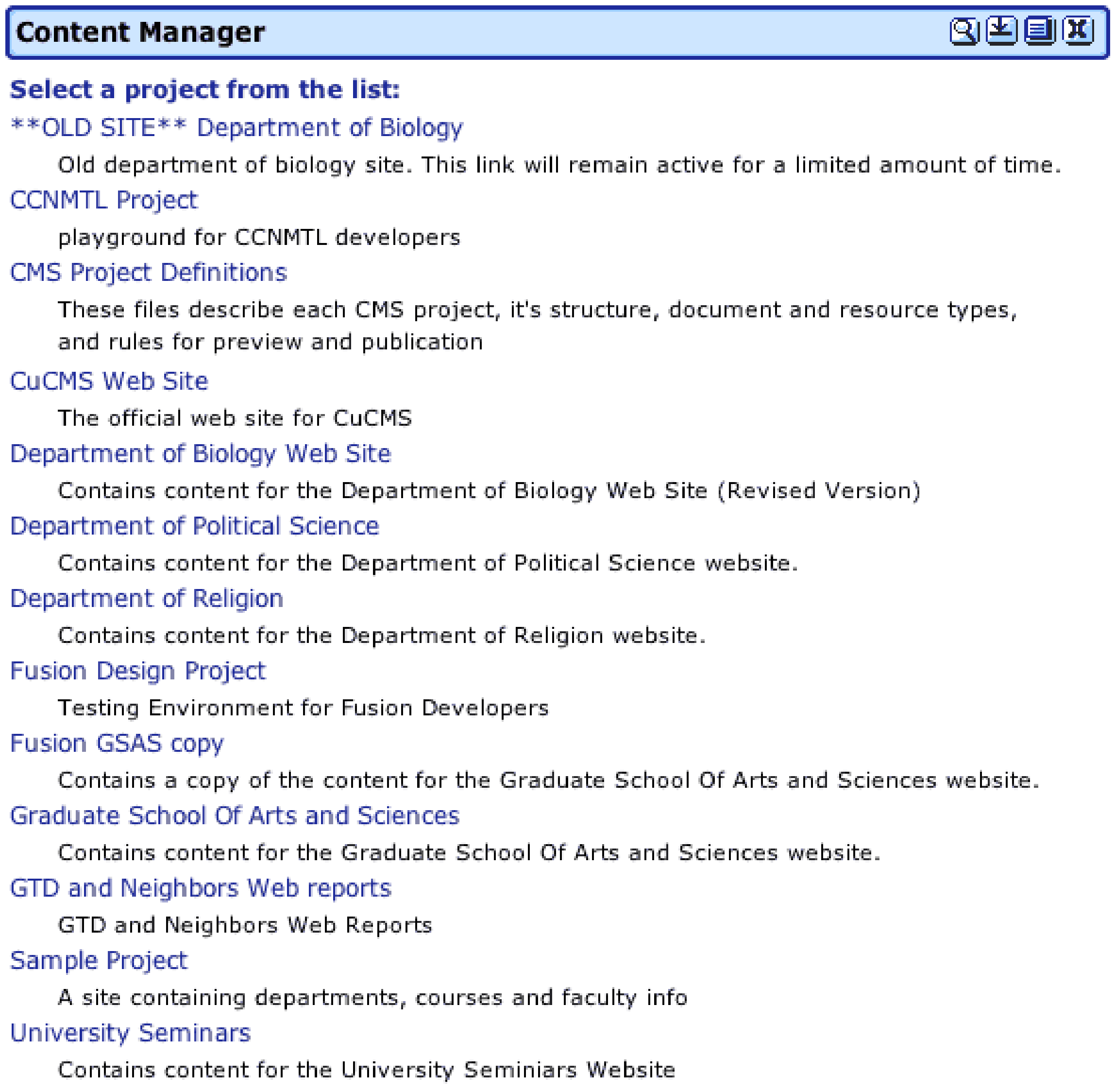
- ***install*** (default) - calls deploy, publish, copyzips and creates cucms build directories under the uPortal web application
- ***publish*** publishes the Content Manager channel into uPortal - requires database access.
- ***deploy*** This target deploys HyperContent media, properties, and libraries into the uPortal web application
- ***copyzips*** copies the bootstrap and sample-project data from zip files into directories under ${runtime.dir}/cucms-repositories/
- ***uninstall*** calls unpublish and undeploy, and deletes cucms repositories and build directories (warning: this will delete any changes you have made to data in these directories!)
- ***unpublish*** This target unpublishes the Content Manager channel from uPortal (requires database access)
- ***undeploy*** This target removes HyperContent media, properties, and libraries from the uPortal web application
- ***compile*** This target compiles the source files under the source directory, into the classes directory.
- ***jar*** This target compiles the sources, copies in stylesheets, and creates a fresh HyperContent jar file

Using the Content Manager

The Content Manager is the primary human interface for HyperContent; it is the entry point for administrators, content authors and developers alike. Content Manager is a uPortal channel; it must be installed, published and subscribed to by a user before it will be visible. Please see the "Getting Started" section for more information about this process.

HyperContent organizes information according to project ; each project represents a unique configuration of content types, structure, underlying filesystems, and authoring environments. Each project might represent a particular web site, or a repository of data of interest to a particular population. There is a special project called "CMS Project Definitions" which is used to store the XML configuration files for every other project in the system; via this bootstrap project new projects can be added, or existing projects altered or extended.

When the Content Manager first appears on screen, it will show you a list of projects to which you have access. If you are an authenticated portal administrator, you will see a list of all projects in the system.



Select a project from the list:

OLD SITE Department of Biology
: Old department of biology site. This link will remain active for a limited amount of time.

CCNMTL Project
: playground for CCNMTL developers

CMS Project Definitions
: These files describe each CMS project, it's structure, document and resource types, and rules for preview and publication

CuCMS Web Site
: The official web site for CuCMS

Department of Biology Web Site
: Contains content for the Department of Biology Web Site (Revised Version)

Department of Political Science
: Contains content for the Department of Political Science website.

Department of Religion
: Contains content for the Department of Religion website.

Fusion Design Project
: Testing Environment for Fusion Developers

Fusion GSAS copy
: Contains a copy of the content for the Graduate School Of Arts and Sciences website.

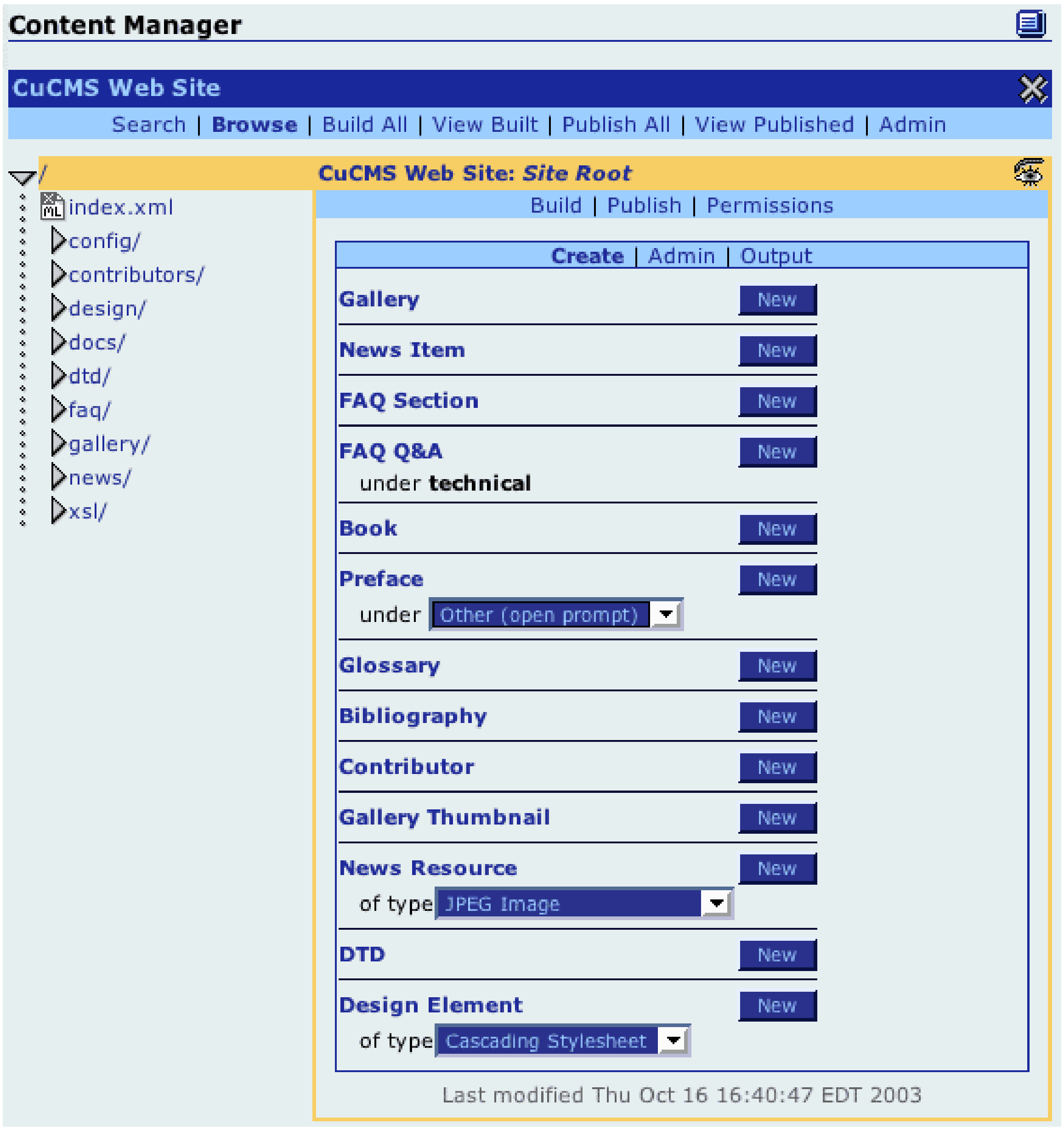
Graduate School Of Arts and Sciences
: Contains content for the Graduate School Of Arts and Sciences website.

GTD and Neighbors Web reports
: GTD and Neighbors Web Reports

Sample Project
: A site containing departments, courses and faculty info

University Seminars
: Contains content for the University Seminiars Website

Click on the title of any project to open it. Once it is open, you will be taken to the project's browse screen. This is divided into three main areas: across the top, beneath the project title, is the top navigation; on the left is the directory tree showing the repository filesystem; on the right is the details screen which allows you to perform any of the available administrative functions, or enter authoring mode. Subsequent chapters describe these different sections of the interface.


Content Manager

CuCMS Web Site

Search | **Browse** | Build All | View Built | Publish All | View Published | Admin

/

index.xml
config/
contributors/
design/
docs/
dtd/
faq/
gallery/
news/
xsl/

CuCMS Web Site: *Site Root*

Build | Publish | Permissions

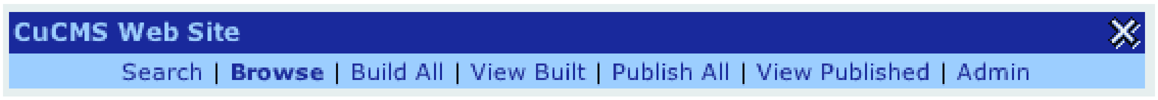
Create | Admin | Output

Gallery	New
News Item	New
FAQ Section	New
FAQ Q&A under **technical**	New
Book	New
Preface under Other (open prompt)	New
Glossary	New
Bibliography	New
Contributor	New
Gallery Thumbnail	New
News Resource of type JPEG Image	New
DTD	New
Design Element of type Cascading Stylesheet	New

Last modified Thu Oct 16 16:40:47 EDT 2003

Top Navigation

Once you have opened a project, across the top of the Content Manager and beneath the project title, is a set of links that give you access to the most important project-wide screens and commands


CuCMS Web Site
Search | Browse | Build All | View Built | Publish All | View Published | Admin

By default, you are taken to the **Browse** screen; the other available screens are **Search** and **Admin**. The **Search** and **Browse** screens are discussed in subsequent chapters; the **Admin** screen gives you access to perform project-wide administrative functions, including rebuilding or optimizing the search index, and setting project permissions.

Four actions are also linked from top navigation;

- **Build All**: This link launched a request to build the entire site in a new window. See the Report Screen section, which explains the screen which comes up.
- **View Built**: This link takes you to the latest build of the site
- **Publish All**: This link publishes the current build.
- **View Published**: This link takes you to the currently published site

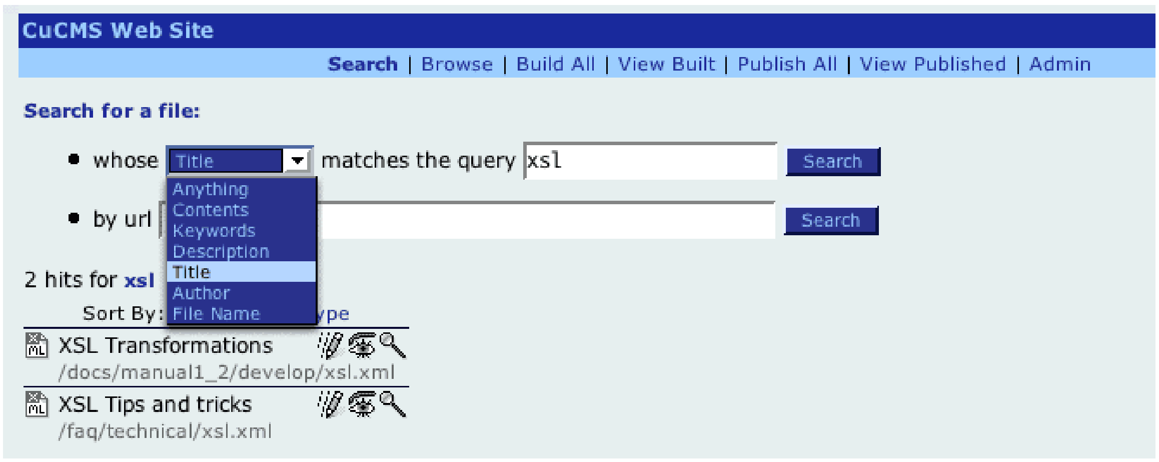
Search

The search screen gives you simple, fast and powerful ways to locate files in the repository, without having to remember their location or browse to them. You can search either by **query** or by pasting in a **URL** from the publish, preview or build site to locate the corresponding file in the repository.

When searching by **query** you can refine the scope of your search by using the provided drop-down list; by default, the scope is set to **Anything**, which searches the complete contents and metadata of every file in the repository, and will also successfully match the first few letters of a file or directory name.

The more specific search scopes are

- **Content** - searches the data of files, but not the metadata or path
- **Keywords** - search the Keywords metadata field
- **Description** - search the Description metadata field
- **Title** - search the Title metadata field
- **Author** - search the Author metadata field
- **File name** - search by complete or partial file name



- whose Title matches the query xsl
- by url

2 hits for **xsl**

Sort By: ...ype

XSL Transformations
/docs/manual1_2/develop/xsl.xml

XSL Tips and tricks
/faq/technical/xsl.xml

Search Results

The results by default are sorted by relevance, but you can also display them by type if, for example, you are specifically looking for images or XSL files.

Each result is presented, on the first line, with an icon representing its content type, it's title or file name, and up to three icons:

- Open the default editor for this file
- Preview this file
- Go to the **Details** screen for this file

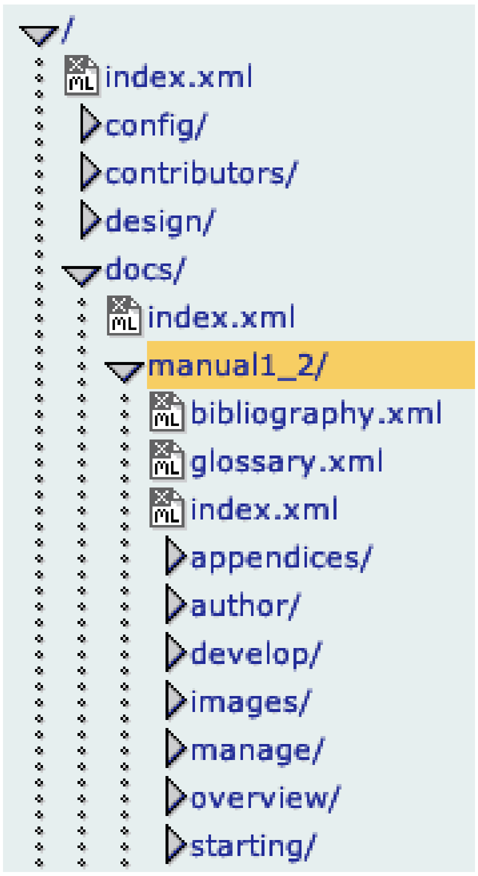
Subsequent lines show the file description, if any, and the file's path in the repository.

Query Syntax

The syntax for search queries in HyperContent is similar to what users might be used to from search engines such as google. If several words are entered in the search box, any results that match one or more of the words are returned, with documents matching more of the terms receiving a higher ranking. To search for a phrase, enclose it in quote marks ("). To assure that a particular word is matched in the search, prefix the word with a "+" sign. To reject any documents matching a particular word, prefix that word with a "-" sign.

Directory tree navigation

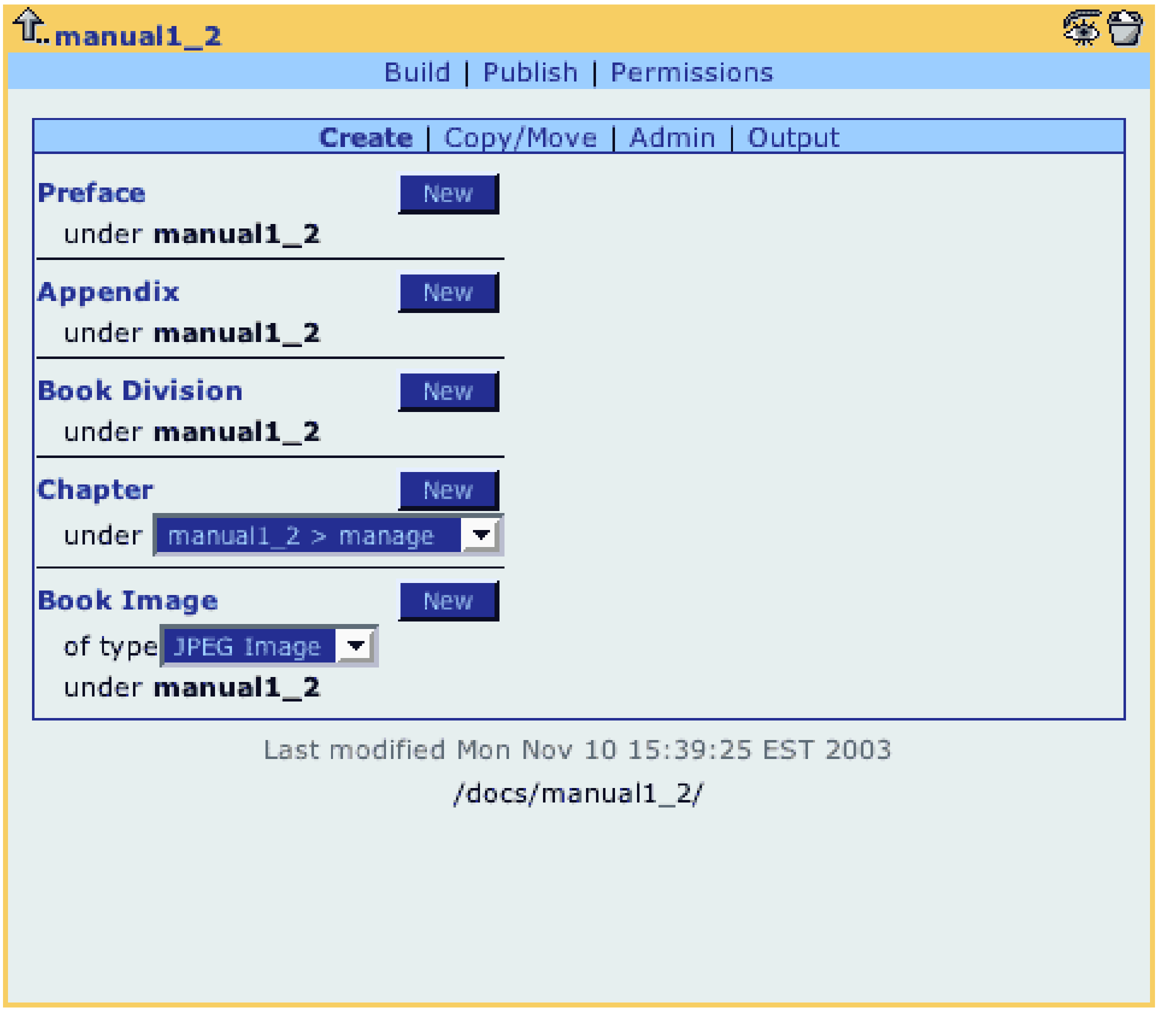
The directory tree component of the Content Manager **browse** screen presents you with a familiar interface to files in the repository. Much like Windows Exporer or the Macintosh Finder, the directory tree presents a hierarchical view of directories that can be expanded and collapsed, and the files which the directories contain.



Clicking on a right-facing arrow next to a directory will expand it to reveal its contents; clicking on a down-facing arrow will collapse that directory and hide its contents. Click on the name of a directory or file to pull up its **Details** screen, described in the next chapter.

Details screen

The Content Manager **Details** screen is reached by clicking on the name of a file or directory in the directory tree, or by clicking in a search results or references list. It appears as a highlighted box to the right of the directory tree, and it refers to details of the highlighted directory or file in the tree.


manual1_2

Build | Publish | Permissions

Create | Copy/Move | Admin | Output

Preface New
under **manual1_2**

Appendix New
under **manual1_2**

Book Division New
under **manual1_2**

Chapter New
under manual1_2 > manage

Book Image New
of type JPEG Image
under **manual1_2**

Last modified Mon Nov 10 15:39:25 EST 2003
/docs/manual1_2/

The highlighted section at top displays the file's title or name, and up to four icons:

- go to the details screen for the parent directory (mouseover to show parent path)
- edit this file; opens the current revision in the default editor
- preview this file; opens a new window
- delete this file or directory

Directly beneath this highlighted section, you will see up to three links that perform actions on the selected file or directory:

- **Build** (opens report screen in new window)
- **Publish** (opens report screen in new window)
- **Permissions** (assign access rights to the file or directory)

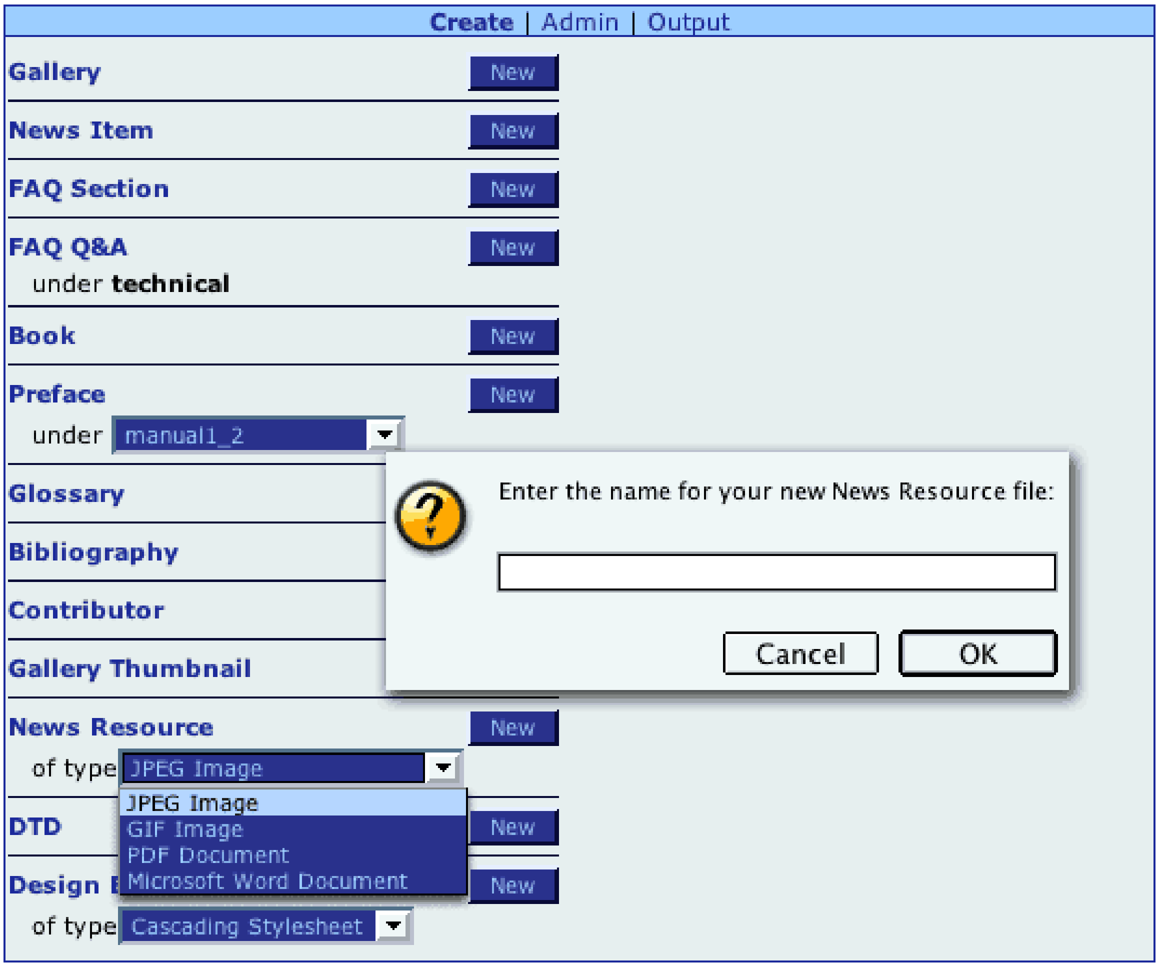
Beneath these links, you might see a file's description, and you may also see a listing for a draft file if you had previously saved a draft, or if a previous editing session had timed out. You can edit, preview or delete the draft using the standard icons which appear next to the listing. Clicking the edit icon from the highlighted row at the top of the details screen opens the current revision, not your last draft; be careful to edit from the draft listing if you want to resume changes previously in progress, otherwise your draft file will be overwritten with a new working copy of the current revision.

Inside the details screen is the action box; this gives access to all the additional features available for the selected node, which may include creating new files, copying or moving the node, viewing references, revisions, or output files. All these additional actions are described in the next few chapters.

At the bottom of the details screen, you will see the last modified date of the node, and the full repository path of the node. This path can be used to reference the selected file from other areas in the site, for example to specify it as an include, link or image in another file.

Create a new file

In the action box on the details screen of the Content Manager, the default section that appears, when applicable, allows you to **Create** new files. You are presented with a list of the document types you can create under the currently highlighted node. In order to create a file in a deeply nested directory, you would normally navigate to and highlight that directory first.



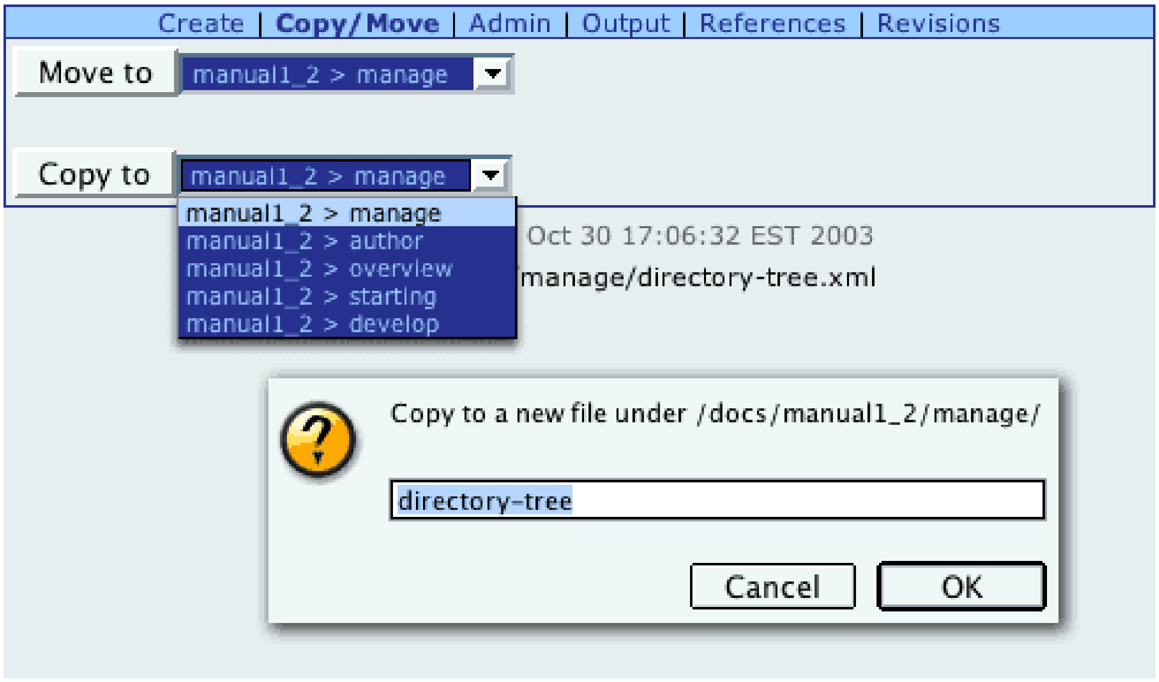
The list contains both XML document types and resource types that can be created in the current context. These various document types are labeled according to the specific project definition; the image above shows document types used for the HyperContent web site. You will see a list specific to the project you are working on; talk to your project administrator for details about the document types used in your project.

If there are several possible paths at which you might create a certain file type, you will see a drop-down list reflecting the options; you may also see a listing "*Other (open prompt)*". Selecting this option and/or clicking the **New** button will prompt you to enter a name for the new directory or file you are creating. For resource files, you may also be presented with a dropdown box you can use to select from one of several configured file types, such as GIF or JPEG when creating an image.

Once you click new, and fill in the prompt if required, you will either get an error if illegal characters were entered in the prompt, or be taken directly to the default editor screen for that file, which is both file type and project specific. Often, the default editor will be the Core Metadata editor.

Copying & Moving files

When applicable, one of the available sections in the action box of the Content Manager details screen allows you to **Copy or Move** the highlighted file or directory.



Oct 30 17:06:32 EST 2003

/manage/directory-tree.xml

Copy to a new file under /docs/manual1_2/manage/

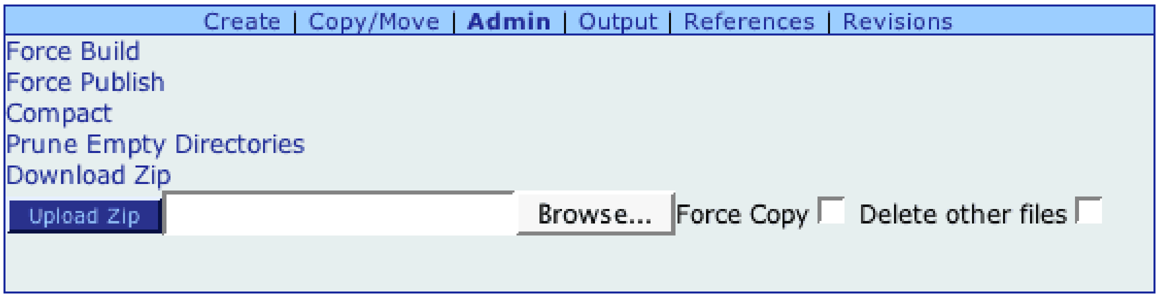
directory-tree

If there is more than one location to which you can copy or move the currently highlighted node, you can choose among them using the dropdown box next to the relevant function. Once you click copy or move, you will be presented with a popup window which allows you to specify the name of the copy, or rename the node if you are moving it. By default, the current name of the highlighted node shows up in the prompt, making it simple to correct minor misspellings, or keep the same name if you are moving the node to a different parent directory.

Note
The options reflected on the copy/move screen are restricted according to your permissions, and will only allow you to copy or move the node to a location in the repository where it will serve the same semantic or structural function as in its current location; thus, for example, you can copy a news file to another news file, but not to a biography page. If you have a legitimate need to copy a file to a location where it will become a different type, you should download the existing file's data, create the new file, and upload the old data into the new file.

Admin section

The admin section of the action box on the Content Manager details screen brings together a few node-specific administration functions.



- **Force Build** whereas a normal build process only updates changed files, a force build will regenerate every output file under this node
- **Force Publish** (same idea)
- **Compact** delete every historical revision of every file under the current node, leaving only the current revision. This is useful for reclaiming disk space, but the history of document changes is lost.
- **Prune Empty Directories** delete all empty directories under the current node
- **Change Directory label** *(not shown)* assign a label to a directory, that will show up in the directory tree to aid users in navigating the site hierarchy.
- **Download Zip** download a zip file containing every file under the current node
- **Upload Zip** upload a zip file with changes to any of the files under the current node

Notes on working with zip files
When you download a zip, each file is archived according to its full path in the repository. Therefore, even if you download a deeply nested directory, when you unzip the file the higher level directories will be present, but no files outside of the selected node will be included.

When you upload a file, it is expected that each file be archived according to its full path in the repository. In this way, for example, you can zip a working copy of the entire repository from your local hard disk, but if you upload it into a specific directory, only files in that directory will be updated.

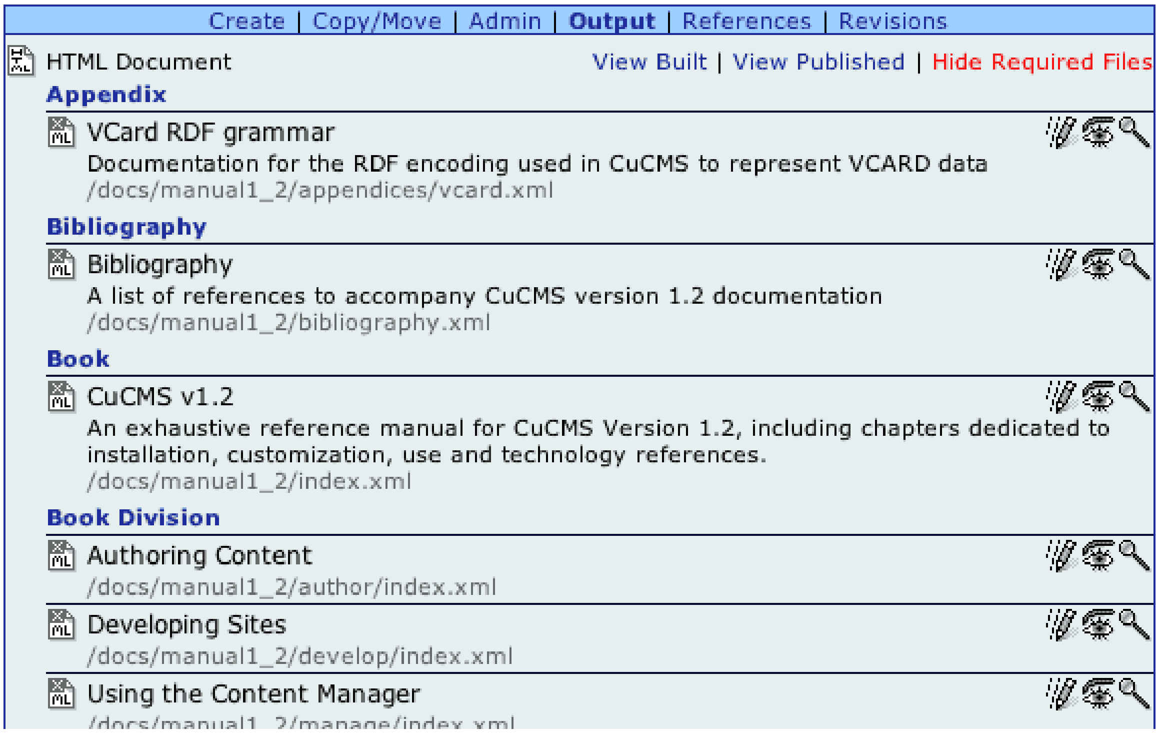
Force Copy
Normally, uploading a zip only saves a file into the repository if the last modified date of the entry in the zip file is later than the last modified date of the file in the repository, or if the file does not already exist in the repository. Force copy will copy all files from the zip under the highlighted node, regardless of date; the overwritten revisions of files that already existed in the repository are still accessible via the revisions section.

Delete Other Files
Normally, files in the repository that are not present in an uploaded zip file are left untouched. If you are interested in synchronizing the repository to your zip file, to bring it in parallel with your working copy or a copy from another server, selecting this option will delete any files under the current node that are not present in the uploaded zip. Use this command with caution, as the deleted files are not recoverable.

Metadata in Zip Files
When you download a zip, each repository file is actually represented by two files in the zip; one has the name of the repository file, and the other is the name of the repository file with ".rdf" appended. This extra file contains the XML encoded RDF metadata for that file; you may edit the metadata manually in this file, though you must be careful to avoid changes that might not be parseable, as metadata must be in a format that the software can understand. If you are uploading new files in a zip, you can leave out the rdf file and an empty metadata set will automatically be generated by the server. However, do not delete the rdf for files you have downloaded and are uploading back to the server, unless you want to lose all of the metadata for those files!

Output section

The output section of the action box on the Content Manager details screen shows you what kinds of output are generated from the highlighted repository file. You can link directly to see the currently built or published versions of each form of output, and you can show the include and XSL files required to generate this output.


Create | Copy/Move | Admin | Output | References | Revisions

HTML Document View Built | View Published | Hide Required Files

Appendix

VCard RDF grammar
Documentation for the RDF encoding used in CuCMS to represent VCARD data
/docs/manual1_2/appendices/vcard.xml

Bibliography

Bibliography
A list of references to accompany CuCMS version 1.2 documentation
/docs/manual1_2/bibliography.xml

Book

CuCMS v1.2
An exhaustive reference manual for CuCMS Version 1.2, including chapters dedicated to installation, customization, use and technology references.
/docs/manual1_2/index.xml

Book Division

Authoring Content
/docs/manual1_2/author/index.xml

Developing Sites
/docs/manual1_2/develop/index.xml

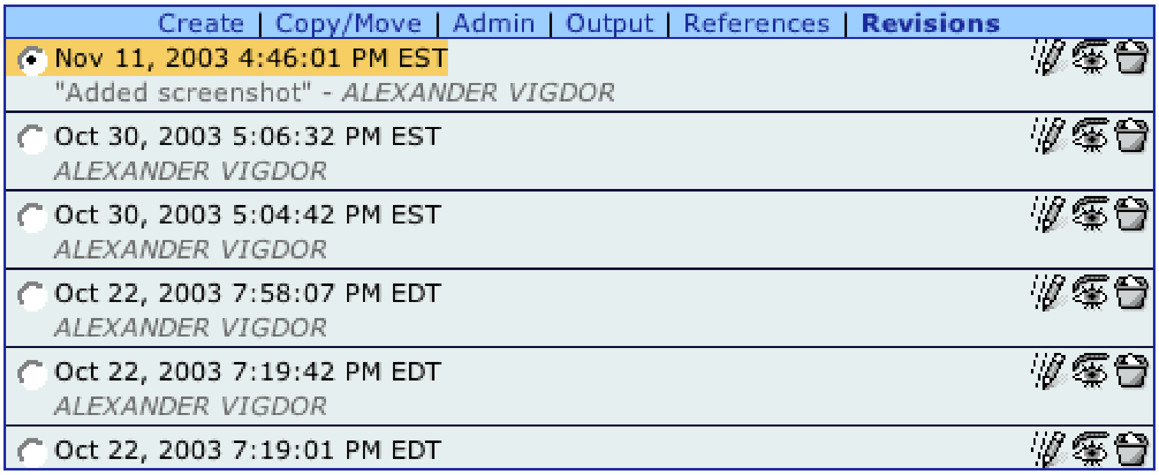
Using the Content Manager
/docs/manual1_2/manage/index.xml

References

The references section of the action box on the Contenent Manager details screen shows you a listing of files which include the currently highlighted file; these are all the files that might be affected by changing or deleting the highlighted file.

Revisions section

The revisions section of the action box on the Content Manager details screen show a complete list of all historical revisions of the currently highlighted repository file, along with who saved that revision and any comments they entered. The current revision is highlighted and marked as selected by a radio button to its left; clicking the radio button next to any other revision will mark it as the current revision, making it very simple to back out of changes made to a file.


Create | Copy/Move | Admin | Output | References | Revisions

Nov 11, 2003 4:46:01 PM EST
"Added screenshot" - ALEXANDER VIGDOR

Oct 30, 2003 5:06:32 PM EST
ALEXANDER VIGDOR

Oct 30, 2003 5:04:42 PM EST
ALEXANDER VIGDOR

Oct 22, 2003 7:58:07 PM EDT
ALEXANDER VIGDOR

Oct 22, 2003 7:19:42 PM EDT
ALEXANDER VIGDOR

Oct 22, 2003 7:19:01 PM EDT

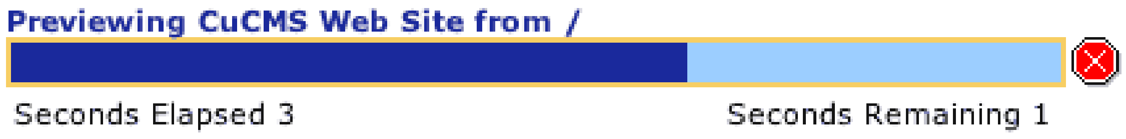
The standard icons to the right of each listing will allow you to

- Open this revision in the default editor for this file
- Preview the output for this revision
- Permanently delete this revision

Report screen

The report screen is launched in separate window from the portal any time a complex action is requested by the user, including preview, build, publish, compact revisions, rebuild search index, download or upload zip.

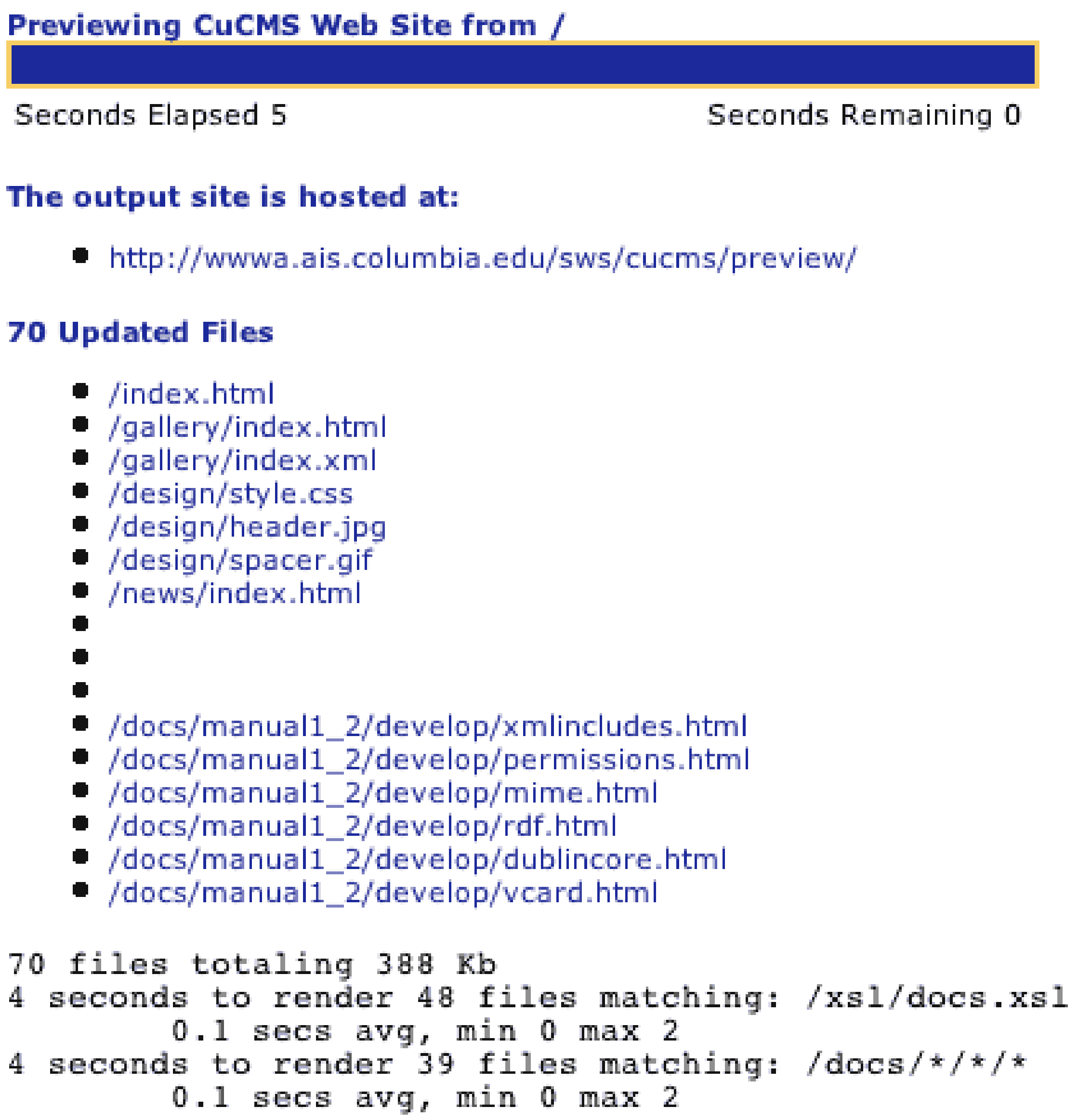
While the action is in progress, only the status bar will appear, giving you both a visual cue and estimate in seconds of how long the action will take to complete.



The stop sign icon on the right allows you to gracefully stop the action in progress; a build or publish command that is stopped will easily be picked up by re-issuing the command, as only files that are not up-to-date are changed in a normal build or publish request. Most requests are composed of many smaller activites, such as building an individual file; the currently active "small" process is allowed to complete normally before the overall request is stopped; only in the case where that small process exceeds a 10 second timeout will it be killed prematurely.

Once the action is completed, sections below the status bar will show you

- where the output is located (if any)
- what files were updated
- what files were deleted
- what errors occured
- any metrics compiled during the request



Previewing CuCMS Web Site from /

Seconds Elapsed 5 Seconds Remaining 0

The output site is hosted at:

- http://wwwa.ais.columbia.edu/sws/cucms/preview/

70 Updated Files

- /index.html
- /gallery/index.html
- /gallery/index.xml
- /design/style.css
- /design/header.jpg
- /design/spacer.gif
- /news/index.html
-
-
-
- /docs/manual1_2/develop/xmlincludes.html
- /docs/manual1_2/develop/permissions.html
- /docs/manual1_2/develop/mime.html
- /docs/manual1_2/develop/rdf.html
- /docs/manual1_2/develop/dublincore.html
- /docs/manual1_2/develop/vcard.html

```
70 files totaling 388 Kb
4 seconds to render 48 files matching: /xsl/docs.xsl
        0.1 secs avg, min 0 max 2
4 seconds to render 39 files matching: /docs/*/*/*
        0.1 secs avg, min 0 max 2
```

Metrics
Each request may have a different set of metrics associated with it, or none at all. For example, a compacting request reports how many old revisions were removed. A rebuild index request reports how many files were indexed.

Preview, build and publish requests have a set of metrics designed to help you identify potential implementation bottlenecks in you site, including

- How many files were processed
- The total number of kilobytes output
- total, minimum, maximum and average processing times for files associated with a

particular XSL stylesheet, or with a particular document types (identified by the pattern associated with that type)

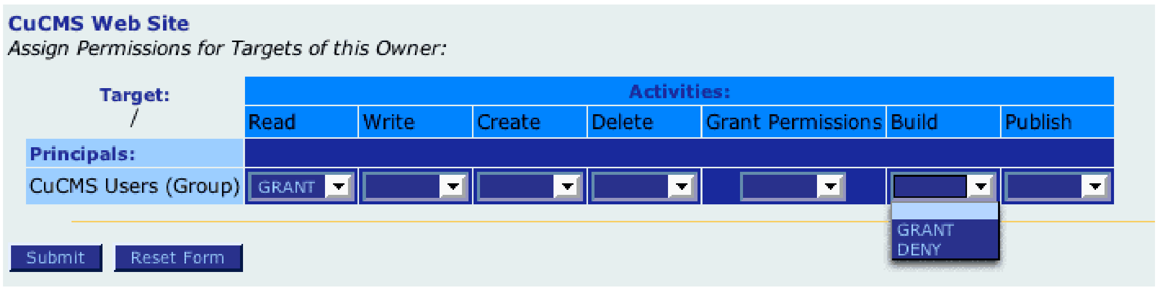
Note that the processing times will not add up to the total seconds elapsed as indicated at the top of the report: various metrics will overlap, since each metric aggregates times according to different but not mutually exclusive criteria. In order to highlight the most important data, a rendering request only displays metrics whose aggregate time exceeds 1 second.

Permissions

Permissions allow you to define who can perform what activites on which portions of a site. There are seven activities defined in HyperContent that can have their permissions set independently:

- **Read** view the target repository path and preview its output
- **Write** edit content and save changes
- **Create** create new directories and files
- **Delete** delete directories and files
- **Grant Permissions** grant permissions on nodes
- **Build** build files
- **Publish** publish files

Permissions are set via the uPortal Permissions Manager, which is accessed either by clicking "Permissions" in the details screen of a directory or file, or from the project admin area.



CuCMS Web Site
Assign Permissions for Targets of this Owner:

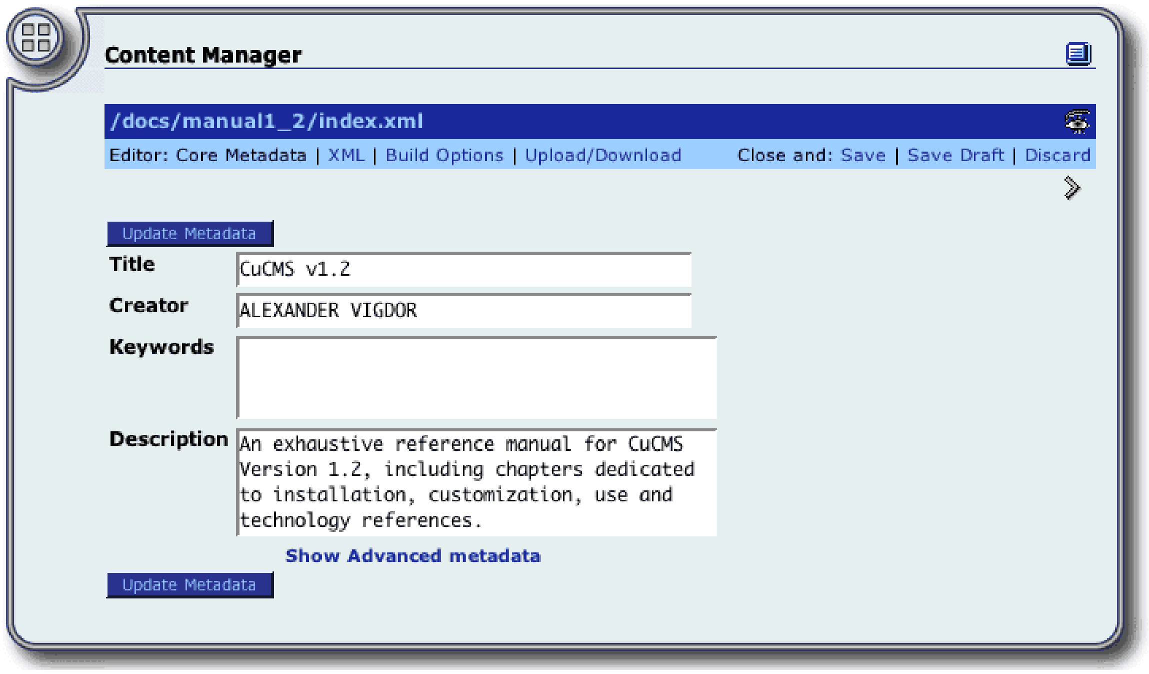
Target: /	Activities:						
	Read	Write	Create	Delete	Grant Permissions	Build	Publish
Principals:							
CuCMS Users (Group)	GRANT						

The target of a permission is the selected repository node at which the permission is granted or denied; this permission then applies recursively to all nodes under the selected node, unless a contrary permission is explicitly set at a lower level. Project permissions are set at the root of the repository; thus, in order to open a project, a user needs to have read permission on the repository root.

The principal you assign permissions to is either a group or person; people inherit the permissions of the groups to which they belong. After clicking Permissions, and before reaching the pictured permissions screen, you will come to a Groups Manager screen which allow you to select the people and groups to whom you would like to grant permissions.

Authoring Content

HyperContent provides a highly extensible and configureable content authoring environment. Authoring functionality is divided into discrete authoring tools, called **editors**, which may be combined in different ways to support different types of content. When a file is opened for editing (by clicking the icon anywhere a file is listed in the Content Manager channel), you enter authoring mode, which looks something like this:



Content Manager

/docs/manual1_2/index.xml

Editor: Core Metadata | XML | Build Options | Upload/Download Close and: Save | Save Draft | Discard

Update Metadata

Title	CuCMS v1.2
Creator	ALEXANDER VIGDOR
Keywords	
Description	An exhaustive reference manual for CuCMS Version 1.2, including chapters dedicated to installation, customization, use and technology references.

Show Advanced metadata

Update Metadata

The path of the file currently being edited is displayed on a dark background across the top. At the right of this area, if applicable, a preview icon is shown; clicking here will preview the output of the current state of your revisions, so you can see the effects of your changes.

On the next line, against a medium background, you are presented with a list of the editors available for this file. Each of the included editors is described in more details in the following chapters. Clicking the name of an editor will switch to that tool, or you can use the « or » buttons to move back and forth between the editors.

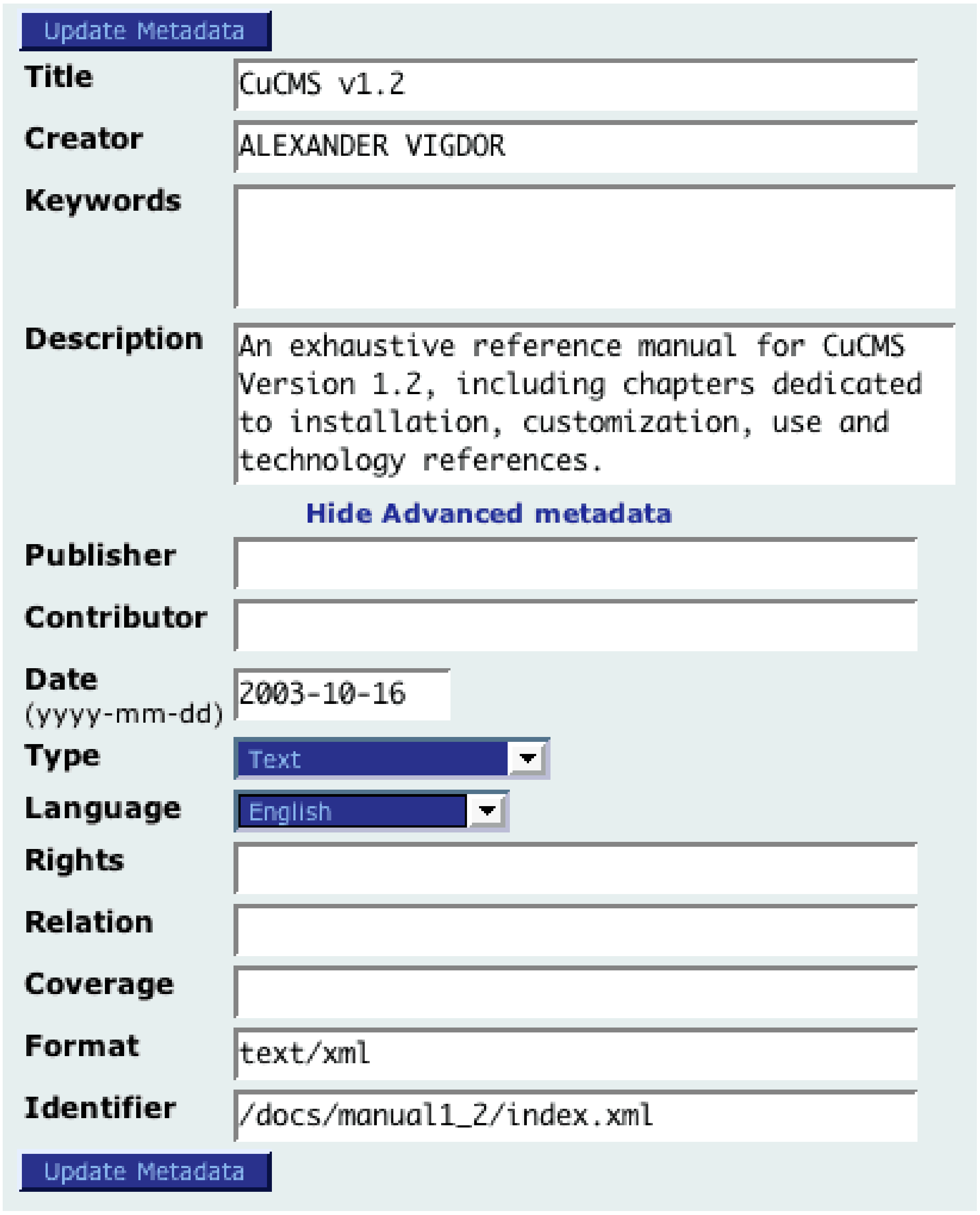
At the right of the medium background area, you see your options for exiting authoring mode:

- **Save** save your changes; the previous revision can still be found on the revisions screen
- **Save Draft** save a draft file of your changes, which you later recover from the file

details screen
- **Discard** throw away your changes

Dublin Core Metadata

The Dublin Core metadata editor allows you to edit the common properties defined by the Dublin Core 1.0 specification. These properties apply to any file type; they are all searchable in the repository, and properties such as title and description are frequently used when building output pages.

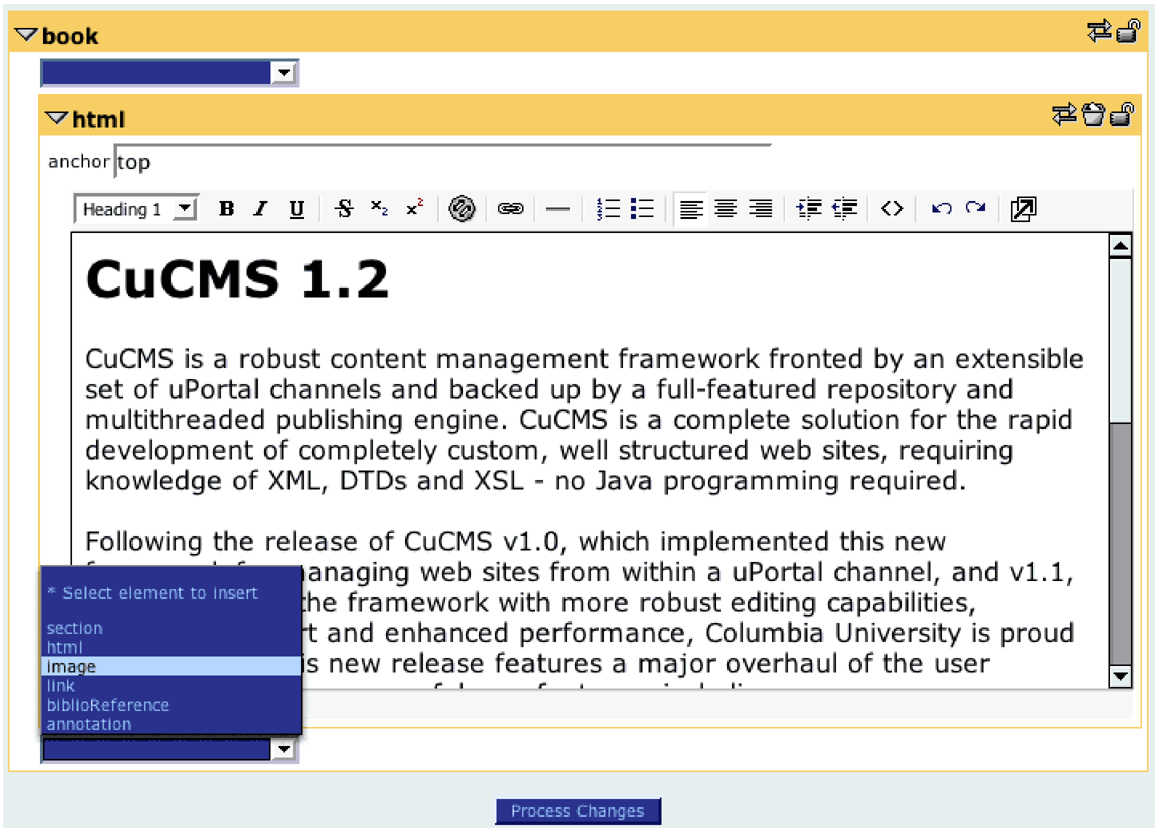


Title	CuCMS v1.2
Creator	ALEXANDER VIGDOR
Keywords	
Description	An exhaustive reference manual for CuCMS Version 1.2, including chapters dedicated to installation, customization, use and technology references.
Publisher	
Contributor	
Date (yyyy-mm-dd)	2003-10-16
Type	Text
Language	English
Rights	
Relation	
Coverage	
Format	text/xml
Identifier	/docs/manual1_2/index.xml

WYSIWYG & XML

The XML/WYSIWYG editor gives you the ability to edit both arbitrary structured data and free-form content with formatted text, lists, images and links.

More complete documentation coming soon



anchor top

CuCMS 1.2

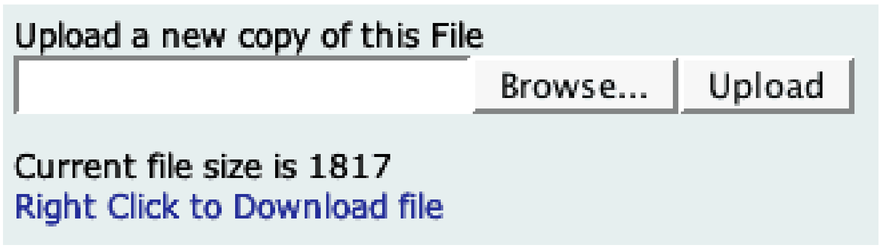
CuCMS is a robust content management framework fronted by an extensible set of uPortal channels and backed up by a full-featured repository and multithreaded publishing engine. CuCMS is a complete solution for the rapid development of completely custom, well structured web sites, requiring knowledge of XML, DTDs and XSL - no Java programming required.

Following the release of CuCMS v1.0, which implemented this new anaging web sites from within a uPortal channel, and v1.1, he framework with more robust editing capabilities, t and enhanced performance, Columbia University is proud s new release features a major overhaul of the user

* Select element to insert
section
html
image
link
biblioReference
annotation

Upload & Download

The upload editor allows you to download data to your desktop, edit it localy and upload your changes.



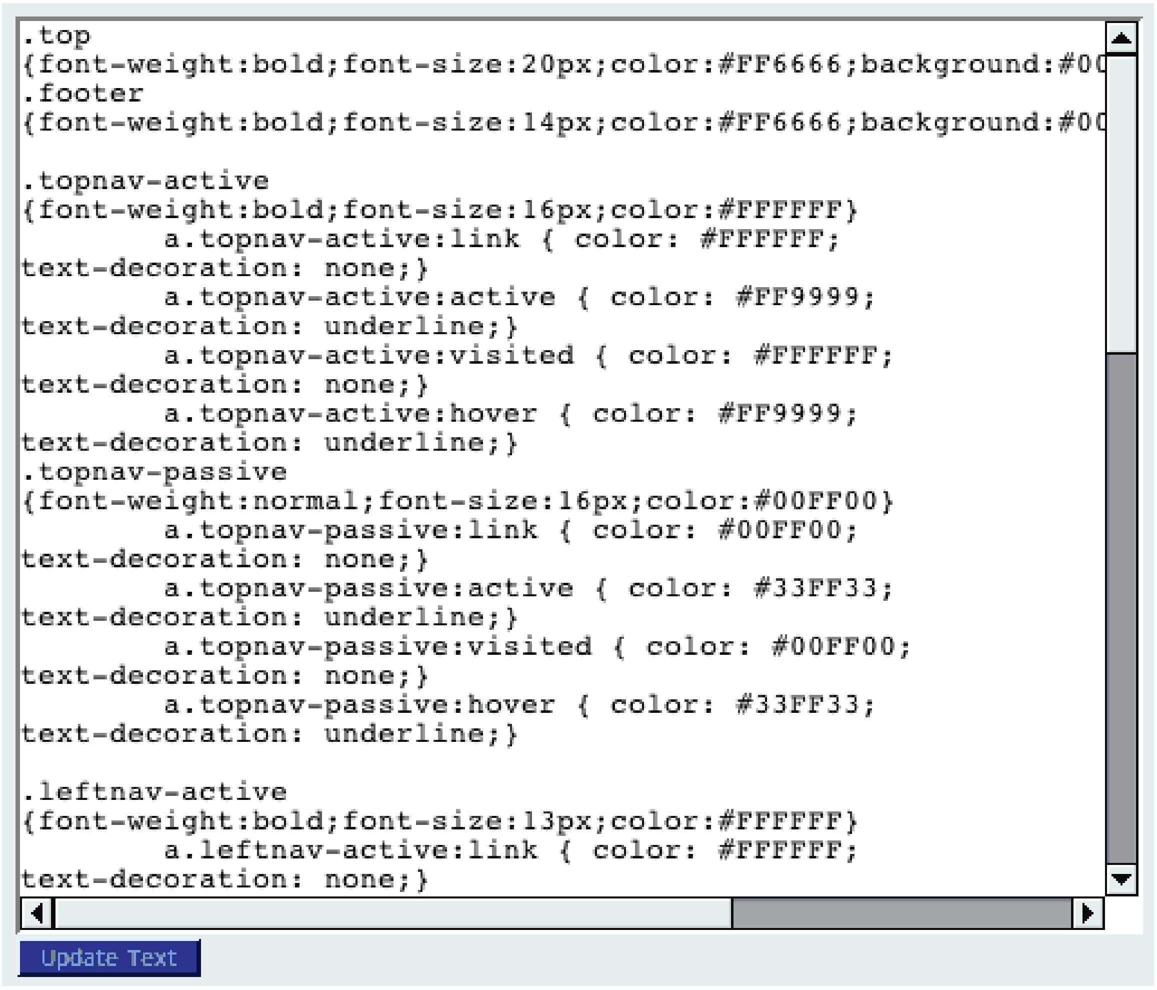
Upload a new copy of this File

Current file size is 1817

Right Click to Download file

Plain Text

The text editor allows you to edit arbitrary text content right in your browser.



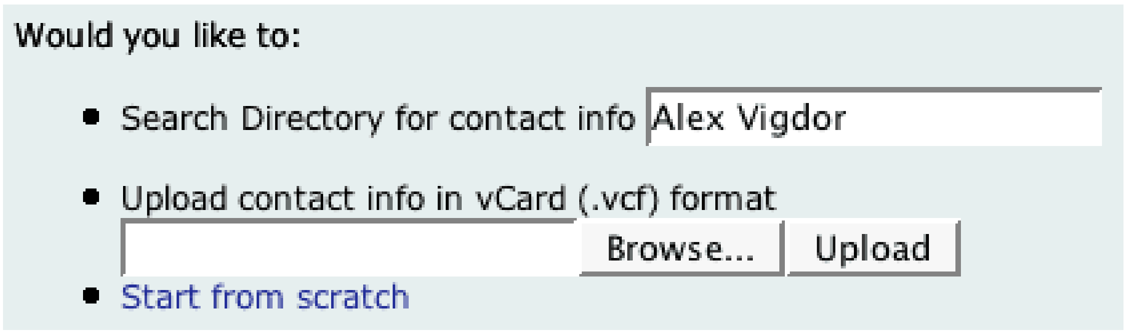
```
.top
{font-weight:bold;font-size:20px;color:#FF6666;background:#00
.footer
{font-weight:bold;font-size:14px;color:#FF6666;background:#00

.topnav-active
{font-weight:bold;font-size:16px;color:#FFFFFF}
        a.topnav-active:link { color: #FFFFFF;
text-decoration: none;}
        a.topnav-active:active { color: #FF9999;
text-decoration: underline;}
        a.topnav-active:visited { color: #FFFFFF;
text-decoration: none;}
        a.topnav-active:hover { color: #FF9999;
text-decoration: underline;}
.topnav-passive
{font-weight:normal;font-size:16px;color:#00FF00}
        a.topnav-passive:link { color: #00FF00;
text-decoration: none;}
        a.topnav-passive:active { color: #33FF33;
text-decoration: underline;}
        a.topnav-passive:visited { color: #00FF00;
text-decoration: none;}
        a.topnav-passive:hover { color: #33FF33;
text-decoration: underline;}

.leftnav-active
{font-weight:bold;font-size:13px;color:#FFFFFF}
        a.leftnav-active:link { color: #FFFFFF;
text-decoration: none;}
```

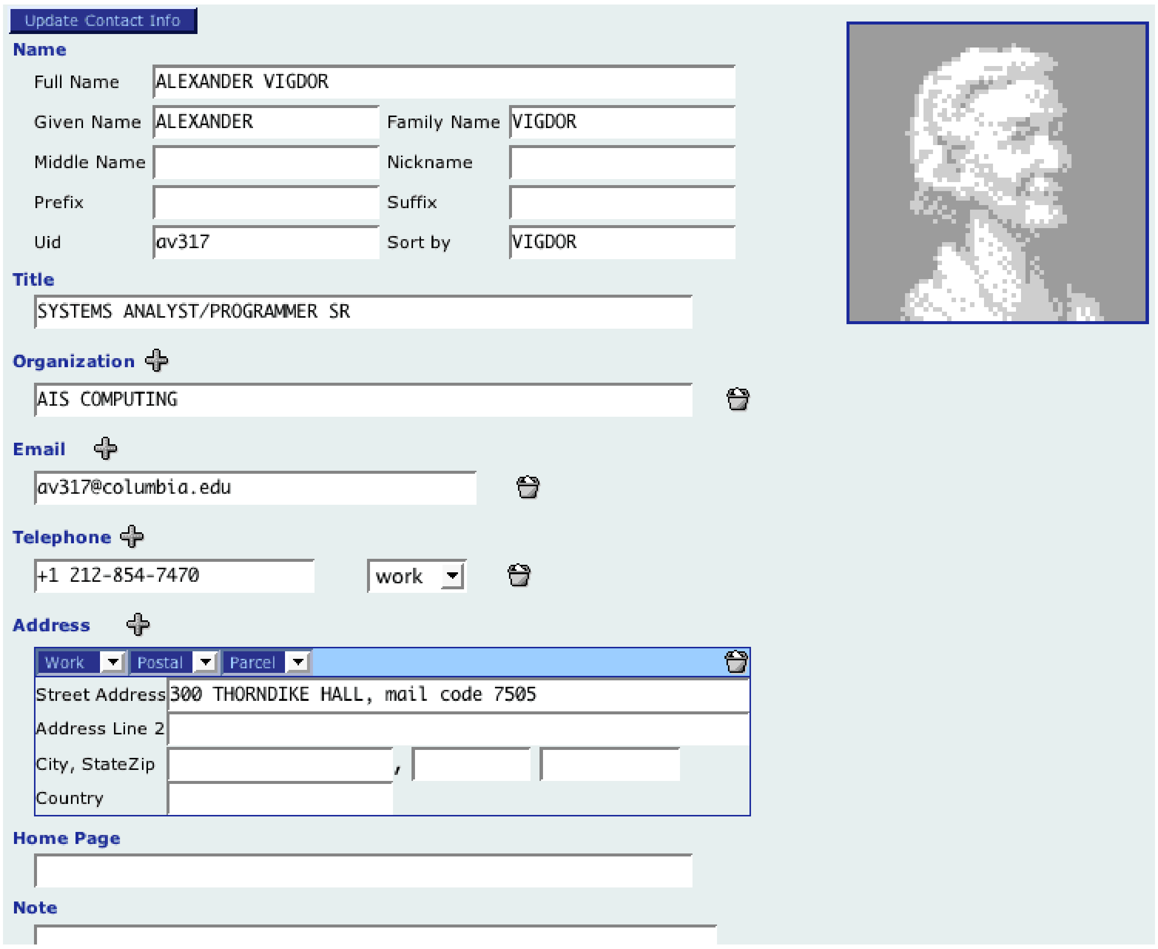
VCard (Contact Info)

The vcard editor allows you to edit contact informaton for people. You can choose to search an ldap directory (if properly configured), upload a vcard file from your desktop Personal Information Manager (such as Outlook), or start a new record from scratch.


Would you like to:

- Search Directory for contact info Alex Vigdor
- Upload contact info in vCard (.vcf) format
- Start from scratch

The contact information page allows you to edit personal data and upload a photo of the person.



Name

Full Name	ALEXANDER VIGDOR		
Given Name	ALEXANDER	Family Name	VIGDOR
Middle Name		Nickname	
Prefix		Suffix	
Uid	av317	Sort by	VIGDOR

Title

SYSTEMS ANALYST/PROGRAMMER SR

Organization

AIS COMPUTING

Email

av317@columbia.edu

Telephone

+1 212-854-7470 work

Address

Work Postal Parcel

Street Address	300 THORNDIKE HALL, mail code 7505
Address Line 2	
City, StateZip	,
Country	

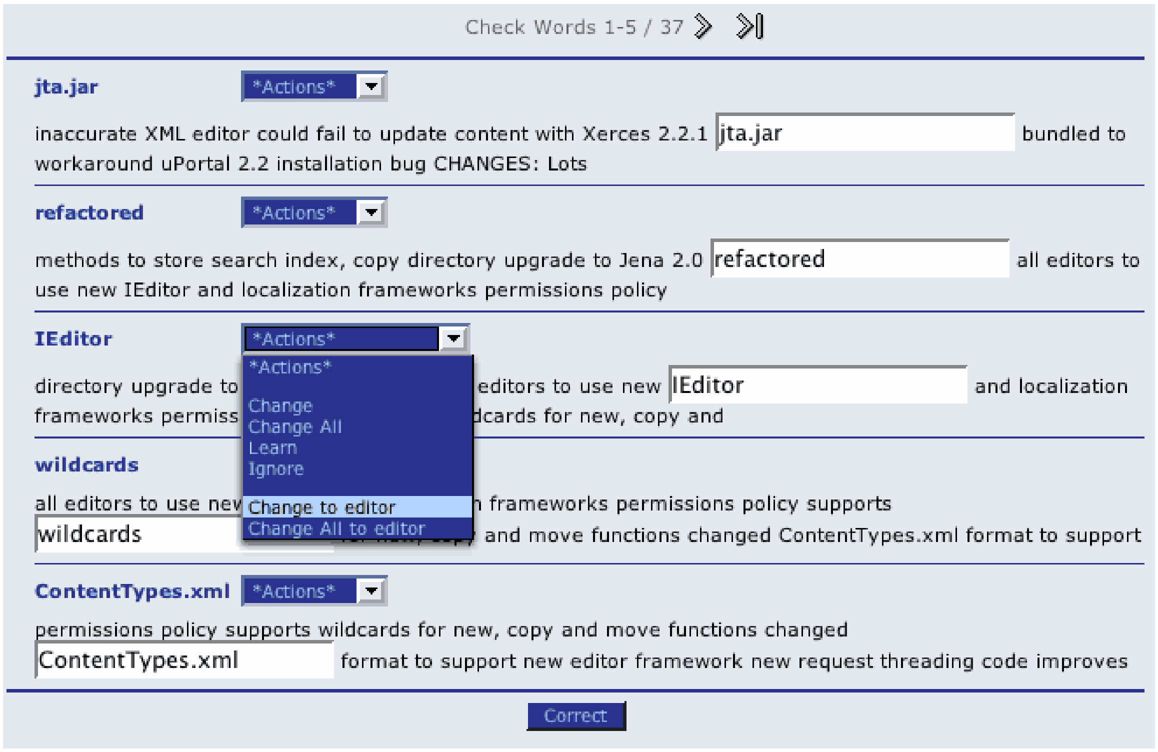
Home Page

Note

Spell Check

The Spell Check editor gives you a simple interface to correct spelling errors in the currently open file. The errors are displayed five at a time, and you can use the controls at the top to page back and forth through the errors.

Each misspelled word is displayed in bold text, followed by an action select box and the word in its original context.



jta.jar

inaccurate XML editor could fail to update content with Xerces 2.2.1 jta.jar bundled to workaround uPortal 2.2 installation bug CHANGES: Lots

refactored

methods to store search index, copy directory upgrade to Jena 2.0 refactored all editors to use new IEditor and localization frameworks permissions policy

IEditor

directory upgrade to editors to use new IEditor and localization frameworks permiss dcards for new, copy and

wildcards

all editors to use nev n frameworks permissions policy supports wildcards and move functions changed ContentTypes.xml format to support

ContentTypes.xml

permissions policy supports wildcards for new, copy and move functions changed ContentTypes.xml format to support new editor framework new request threading code improves

You may type a correction directly into the text box that shows the word in context, or you can check the action list for suggested replacement values. You may also choose to ignore a certain word for the duration of this spell-checking, or if your project is configured with a custom dictionary, you can ask the system to learn the word so it will not be identified as an error in any other documents in the project. You must click the "correct" button for your changes to take effect.

Multilingual support

By default, the spell checker analyzes your document against the best dictionary it can find corresponding to your session locale (user configurable in uPortal 2.2). You can explicitly change the language of a specific document using the Dublin Core Metadata Editor, and the spell checker will switch to the best available dictionary corresponding to that language. The included dictionaries in version 1.3 of HyperContent are American English (en_US), British English (en_GB), French (fr) and German (de). American English is set as the default if no better match is found.

This feature was introduced in HyperContent v1.3

Images

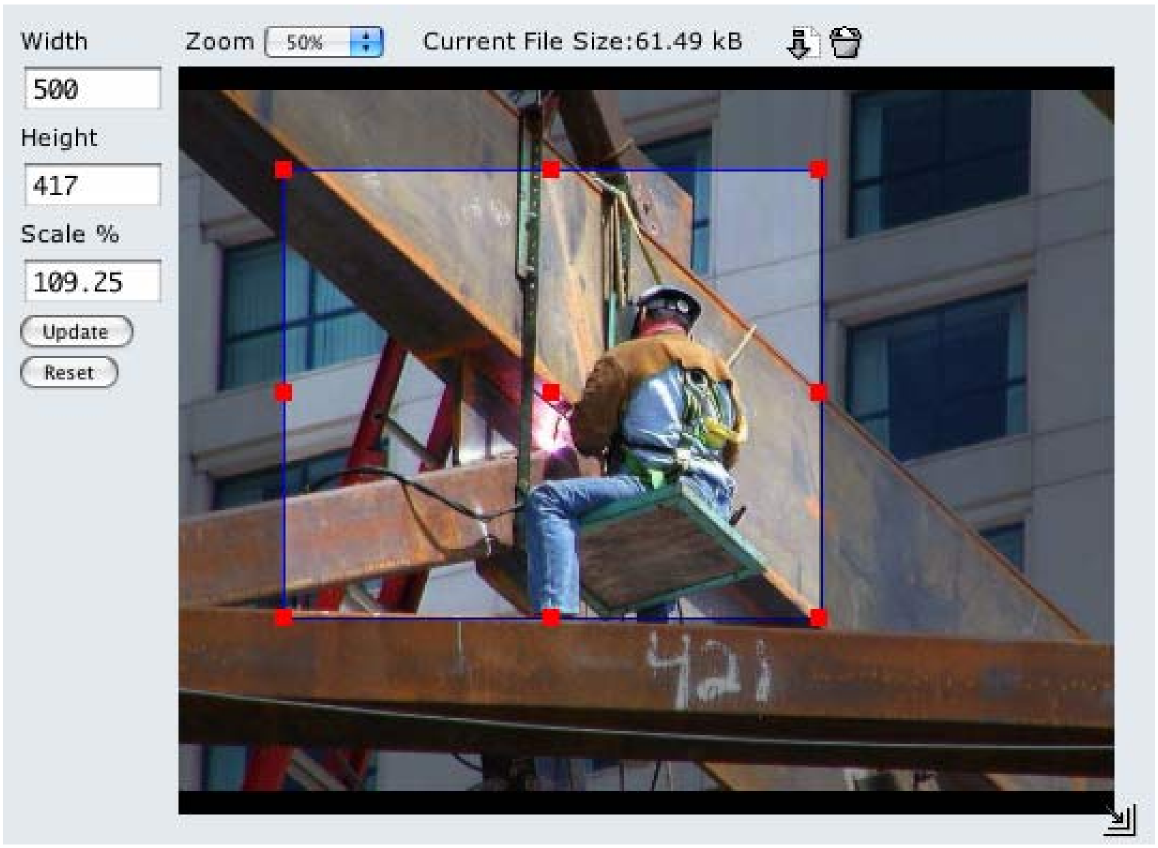
The Image Editor allows you to upload and manipulate photos and images directly in your browser. When you first open the Image editor, if there is not currently any image data in the file, it will present you with an upload form. Once an image is loaded, if you want to load a new image you can delete the current data with the (delete) icon to return to the upload screen. You can use the (download) icon to download the current image data to you local machine.

While working with an image, you can use the 'Zoom' select box to change the level of magnification of the image in your browser. This will have no effect on the resolution of the actual image data, which will always correspond to the '100%' zoom.

The Image editor allows you to perform three common image editing operations: cropping, resizing and converting. Conversion happens automatically: if you are editing a JPG file and upload GIF or PNG data, it will be converted. Similarly, if you are editing a GIF file and upload JPG data, it will be downsampled to 256 colors and converted to GIF format. Conversion is automatic in any direction between GIF, PNG and JPG formats. Other formats may be supported in the future.

Cropping

Cropping is the act of trimming the borders of an image, normally to tighten the frame on the area of interest. You simply click and drag over the image to specify your desired cropping, which is represented by a blue box with red handles. The area which will be trimmed off, outside the blue box, is darkened. You can click anywhere in the darkened area to remove the cropping box or start over. You can also click and drag any of the red handles on the blue box to move just that corner or edge, and the red handle in the middle allows you to move the crop box over the image while maintaining its current dimensions. On the left, the Width and Height boxes will update to reflect the width and height of the cropped area, so you can easily adjust your cropping box to specific dimensions.



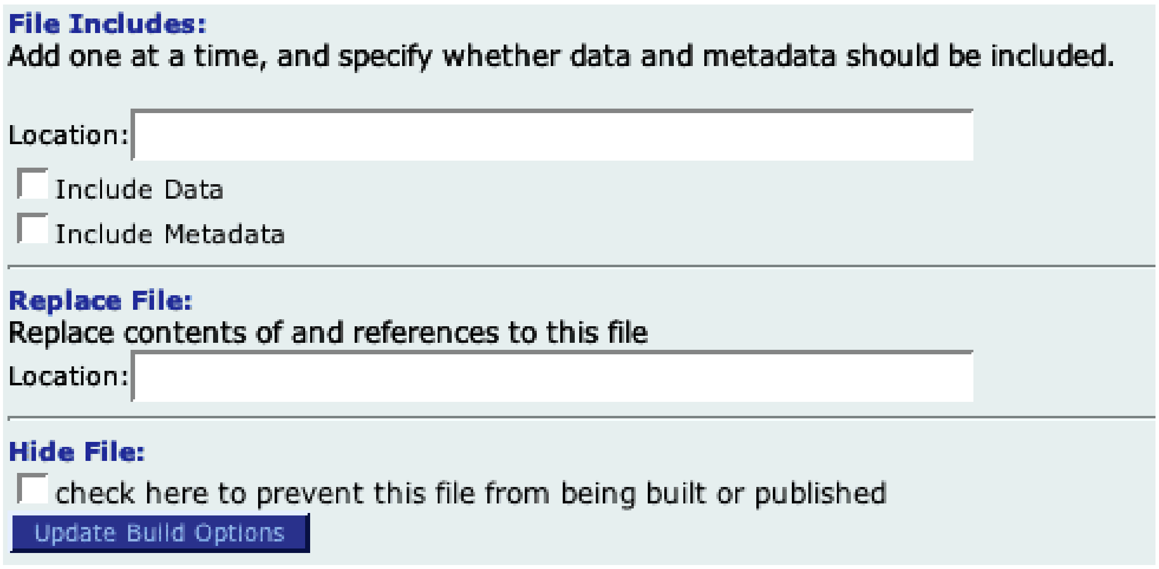
Resizing

Resizing is the act of increasing or decreasing the resolution of an image. For example, if you have uploaded a multi-megapixel image from your digital camera, you normally want to decrease its resolution for display on the web. There are two ways to control the resizing of your image; you can click and drag the ⇲ (resize) icon on the lower right corner of the image to increase or decrease the resolution, or you can type in a target width, height or percentage scale in the boxes along the left side of the image. Adjustments using any of these controls will update the others automatically, and if you have a cropping box, it will be resized along with the image so that it continues to frame the same portion of the image.

This feature was introduced in HyperContent v1.3

Build Options

The Build Options editor allows you to specify some file-specific parameters that will be used at the time the file is built.


File Includes:
Add one at a time, and specify whether data and metadata should be included.

Location:

☐ Include Data

☐ Include Metadata

Replace File:
Replace contents of and references to this file

Location:

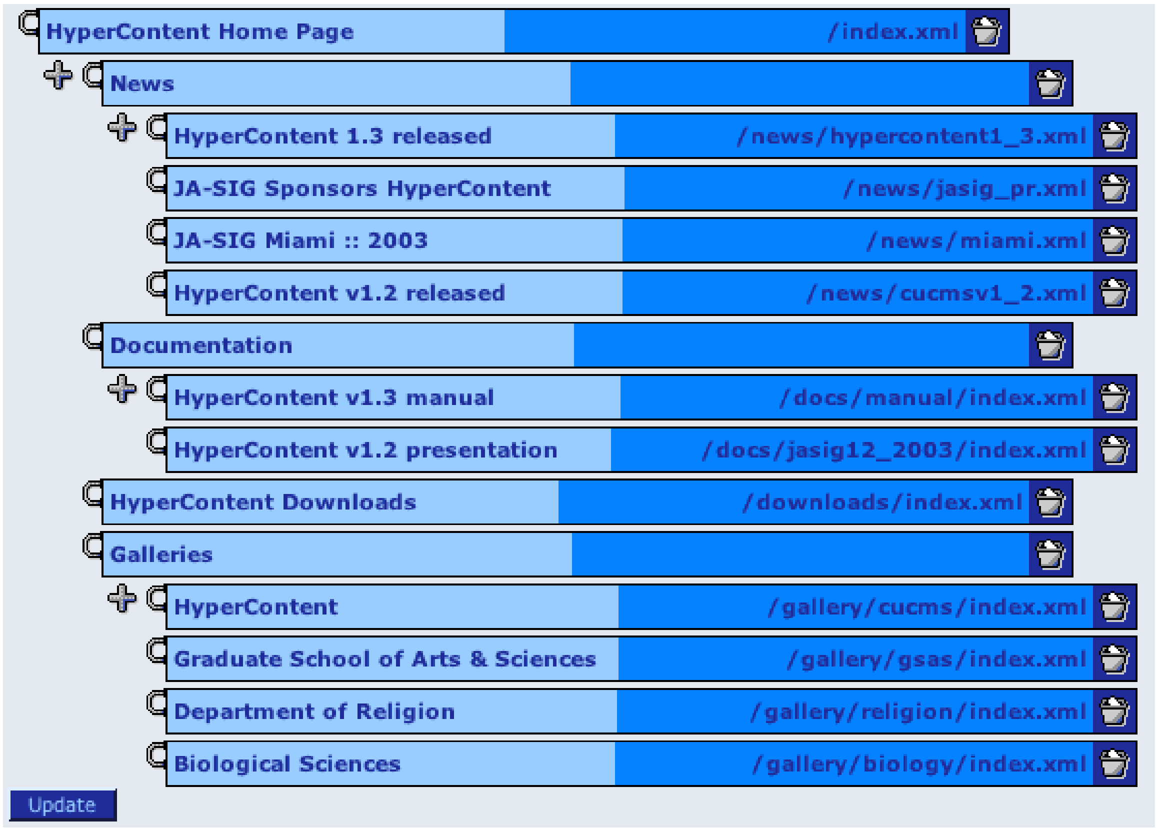
Hide File:

☐ check here to prevent this file from being built or published

One option is to add includes; for example, by including a particular faculty member bio in a course page, it might allow the site to generate links between the course and faculty member. Replacement allows you to tell the system to completely replace the contents of a file with those retrieved from another file in the repository, or even from a URL. This makes it possible, for example, to link to some source of XML on the internet and have it fulfill a particular function within your site. You can also "hide" the file, such that any output versions of the file, and any references to it, will be removed from the next site build.

Navigation / Site Map

The navigation editor allows you to easily shape navigation or site map information based on templates configured in the XML editor. You can re-order items within a list by clicking and dragging the handle icon to the left of a listing, suppress an item by clicking the trash can icon, or add a new link at a given level by using the add icon.



HyperContent Home Page	/index.xml
News	
HyperContent 1.3 released	/news/hypercontent1_3.xml
JA-SIG Sponsors HyperContent	/news/jasig_pr.xml
JA-SIG Miami :: 2003	/news/miami.xml
HyperContent v1.2 released	/news/cucmsv1_2.xml
Documentation	
HyperContent v1.3 manual	/docs/manual/index.xml
HyperContent v1.2 presentation	/docs/jasig12_2003/index.xml
HyperContent Downloads	/downloads/index.xml
Galleries	
HyperContent	/gallery/cucms/index.xml
Graduate School of Arts & Sciences	/gallery/gsas/index.xml
Department of Religion	/gallery/religion/index.xml
Biological Sciences	/gallery/biology/index.xml

this feature was introduced in HyperContent 1.4

Developing Sites

This developer documentation seeks to describe the fundamental building blocks and choices which go into creating a completely custom web site with HyperContent.

The code base for HyperContent has been developed with the philosophy of attempting to accommodate the unforeseen; there are no assumptions made about the structure of a web site, of information or navigation architecture, supported media types or data structures. There is an XML grammar defined for specifying the directory structure of a web site and defining certain semantics and relationships of components within that structure. At the most granular level of data, the built-in-tools rely on XML DTDs to define data structures, and RDF to support metadata structures. The generation of HTML or PDF can be accomplished through built-in support for XSL transformations, while a filtering architecture leaves open the possibility of plugging in other templating languages. Data reuse across multiple pages is accomplished through the use of XML Includes. Built-in authoring tools support a broad number of applications, while a framework makes it easy to plug in additional authoring tools and mix and match different authoring tools as required in a given project.

The tradeoff for all of this flexibility is that developing a new site from the ground up can be a bit daunting, as there are many configuration choices that can be made, and much thought must go into the development of DTDs to define data structures and the use of XML includes and XSL to automate construction of site navigation. As you get started developing with HyperContent, you may want to read through this and build up a project as you go. Once you are fluent with the process, you will likely still find it useful to refer back to these pages for useful tidbits.

Project Definition Files

HyperContent allows you to configure an unlimited number of projects, each of which can represent a completely unique site. The configuration for a project is encapsulated in its project definition file, which is an XML file format; the contents of this file are discussed throughout the course of this developer documentation, intermixed with the background information and global configuration options you need to understand to make the most of HyperContent.

Project definition files are managed via a special project called "HyperContent Project Definitions", which allows you to use HyperContent to manage its own configuration files. The project definition for this special project is stored in the *Projects.xml* properties file.

This diagram shows the skeletal structure of a project definition file: each highlighted label refers you to the chapter of this documentation that deals with the set of tags directly beneath it. Chapters not listed in the diagram contain more general background, and do not deal directly with the semantics of project configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cms-project SYSTEM "http://hypercontent.sourceforge.net/hypercontent1.3.dtd">
<cms-project>

    <!-- Filesystems -->
    <repository ... />
    <preview ... />
    <build ... />
    <publish ... />

    <!-- XML Content -->
    <xml-doc ... />
    <xml-doctype ... >
        <output ...>

            <!-- XML Includes -->
            <include ... />

            <!-- XSL Transformations -->
            <transform ... />

            <!-- Filters -->
            <filter ... />
        </output>

        <!-- Editors -->
        <editor ... />
    </xml-doctype>

    <!-- Non-XML Content -->
    <resource-directory ... >

        <!-- Content Types -->
        <content type="..." />
    </resource-directory>

    <!-- Ignored Directories -->
    <ignore-directory .../>
</cms-project>
```

Filesystems

The first consideration to be made when sitting down to implement a web site with HyperContent is where will the data live, and where will it end up? HyperContent relies on a single FileSystem API to communicate with both the data repository and the destination for output. Multiple implementations of this API are included; the standard repository implementation uses a locally mounted filesystem and stores data, metadata, and historical revisions of files in a custom structure, as well as providing locking and full content searching. A local non-editioning version is available to publish standard data files to a locally mounted filesystem, and FTP and SFTP implementations make it possible to push content to a filesystem on a remote server, for example if your web servers are on different machines than HyperContent.

Up to four filesystems are configured for each project; a filesystem must be configured for the repository, and if the project is intended to produce output, preview, build and publish filesystems should also be configured. The preview space is for volatile renderings of content that may not be intended for publication. The build space is where current data is rendered to, and can be reviewed for accuracy before being published. The publish space is where built files are copied to in order to make them public. The preview, build and publish filesystem typically map into the document tree of a web server, so that their contents can be viewed with a browser.

Here is a sample configuration XML fragment that shows the four filesystems configured using the four main included implementations, with comments indicating what each of the arguments is.

```
<repository>
    <filesystem-def name="Project Repository" type="local-editioning">
        <!-- local filesystems require a single argument
             that points to the local directory which will serve
             as the logical root of the repository -->
        <argument index="0" value="/repositories/project-repositories/" />
    </filesystem-def>
</repository>
<preview base-url="http://appserver.project.org/project/preview/">
    <filesystem-def name="Project Preview Server" type="local">
        <argument index="0" value="/www/docs/project/preview/" />
    </filesystem-def>
</preview>
<build base-url="http://build.project.org/project/build/">
    <filesystem-def name="Project Build Server" type="ftp">
        <!-- for FTP and SFTP 4 arguments are required -->

        <!-- the hostname -->
        <argument index="0" value="build.project.org" />
        <!-- the username (or prompt for a username) -->
        <argument index="1" prompt="username" type="text" />
        <!-- the password (or prompt for a password) -->
        <argument index="2" prompt="password" type="password" />
        <!-- the directory on the remote server that will act
             as the logical root of the filesystem -->
        <argument index="3" value="/www/docs/project/build/" />
        <!-- the desired permissions state of stored files -->
        <argument index="4" value="0775" />
    </filesystem-def>
</build>
<publish base-url="http://publish.project.org/project/">
    <filesystem-def name="Project Publish Server" type="sftp">
        <argument index="0" value="publish.project.org" />
        <argument index="1" prompt="username" type="text" />
        <argument index="2" prompt="password" type="password" />
        <argument index="3" value="/www/docs/project/" />
        <argument index="4" value="0775" />
    </filesystem-def>
</publish>
```

Path patterns

Every file in a HyperContent project maps to a path pattern specified in one of three main element types in the project definition document: *xml-doc*, *xml-doctype* or *resource-directory*. These elements are discussed in more detail in the "XML Content" and "Non-XML Content" chapters; for now, it is important to note that each of these elements configures a unique class or instance of content, by mapping one path pattern to a semantic label and 0 or more forms of output. A path pattern might simply look like a literal path in the filesystem, such as

```
/index.xml

/config/navigation/topnav.xml
```

or, it might contain wildcards

```
/images/*

/docs/*/*

/courses/*/index.xml
```

When a wildcard occurs as the final character of path pattern, it represents a file; the file extension is not part of the pattern, as it will depend on the element which is being used to configure the content. When a wildcard is followed by a forward slash, it represents a directory name.

Every path pattern for a configured element must begin with a "/" to indicate the root of the repository; patterns are also used to specify includes, covered in a separate chapter, which is the only time you may leave off the leading "/" to specify a relative location. It is important that any paths representing directories end with a trailing "/", such as the base-url of an output filesystem (e.g. "http://hypercontent.sourceforge.net/docs/").

A forward slash "/" should always be used as the directory separator, regardless of the operating system in use.

An Example

Say you want to define a Document Type "Faculty Bio"; you could specify its path at "/faculty/*". Then your repository could look like this:

```
/faculty/

    marybprof.xml
    johnqteacher.xml
```

If you wanted to also have a Document type "Faculty Course Listing" associated with the professor, you would want to organize them together in a directory. In that case, you would want the directory name to be the wildcard:

```
/faculty/*/bio.xml
/faculty/*/courses.xml
```

Which could make your repository more like this:

```
/faculty/

    marybprof/

        bio.xml
        courses.xml


    johnqteacher/

        bio.xml
```

If you wanted to construct the output with links to a number of individual course pages, also associated with the professor, you could add a "Faculty Course" Document type with a path of "/faculty/*/*". This would allow you to create new courses only for faculty members that have already been created in the project. Your repository could then look like:

```
/faculty/

    marybprof /

        bio.xml
        courses.xml

        m101.xml

        m102.xml


    johnqteacher /
```

```
            bio.xml
```

If you wanted to allow each professor to store graphics to present in their course documents, you could create a resource directory called "Faculty images" with a path of "/faculty/*/images/*" accepting content types image/jpeg and image/gif.

```
    /faculty/

        marybprof /

            bio.xml
            courses.xml

            m101.xml

            m102.xml

            images/

                m101Logo.gif
                m102Logo.jpg

                myportrait.jpg

        johnqteacher /

            bio.xml
```

In the Content Manager, users will only be presented with options to create new files in locations where only a single wildcard is left to fulfill. For example, you could not create a new "Faculty image" file at "/faculty/jimassistant/images/jimbo.jpg" without first creating "/faculty/jimassistant/bio.xml" or "/faculty/jimassistant/courses.xml".

As you can imagine, this flexible system of defining paths can accommodate very deep and complex filesystems while maintaining a strict understanding of what kinds of data live in each directory. This controlled directory structure is essential to the power of XML includes, as data relationships are imprinted onto the very structure of the filesystem.

Content Types

HyperContent is capable of managing any type of file that has an associated MIME type. MIME types are globally configured in the properties file "ContentTypes.xml": this file must contain a declaration for any MIME type you want to use in a HyperContent Project. Here is a sample content type declaration:

```
<content-type type="image/jpeg" label="JPEG Image"
              icon="media/edu/columbia/hypercontent/doc_jpg.gif">

    <equivalent type="image/pjpeg"/>

    <file-extension value="jpg" preferred="true"/>
    <file-extension value="jpeg"/>
    <file-extension value="jpe"/>

    <editor key="image"/>
    <editor key="dublincore"/>

</content-type>
```

The root level tag of a content-type declaration, "content-type", has three attributes:

- **type**: either the preferred or only mime type for this content
- **label**: human-readable name for an instance of this content type
- **icon**: the location of a file (relative to the web application root) representing an icon for this content type
  (a generic blank icon is provided at "media/edu/columbia/hypercontent/doc_blank.gif")

There are four tags which may appear inside a content-type declaration:

- **equivalent**: specifies an alternate mime type for the same content
- **file-extension**: specifies a file extension associated with this content type. If there is more than one, one should have the attribute **preferred="true"** to indicate it is the preferred extension.
- **editor**: the key of an authoring tool, as specified in Editors.xml, which should be a global default for this content type. Multiple editors can be configured, and will appear in the order listed. Default editors can be overridden within a project's definition file.
- **word-loader**: *(not shown)* has a single attribute "class" used to specify an implementation of the IWordLoader API that can be used to expose a file's content to the spell checker.

Content types are used to configure the contents of resource directories in HyperContent projects; see the chapter "Non-XML Content" for more information.

XML Content

A Brief Introduction to XML

XML is a text format that allows for information to be expressed inside of tags that describe its structure. What makes XML useful is the preponderance of tools available for accessing and manipulating this structured data, from libraries in all the major programming languages to built-in functions in major web browsers.

Grammars

A related set of XML tags, their possible structural permutations and data contents constitute an XML grammar. XML grammars can be defined by anyone to meet their own needs, or can be agreed upon by communities to enable a standardized exchange of like information between disconnected and even competitive products. Grammars can be described in a DTD (Document Type Definition) or XML Schema file, enabling standard processing tools to determine whether an XML document is a valid instance of a particular grammar.

Transformations

A powerful companion technology to XML is called XSL: it provides a way to transform XML from one grammar to another using templates. It is this technology which makes XML perfectly suited to managing web site content; data can be stored in XML according to custom grammars, and XSL makes it possible to output that data as XHTML for viewing in a web browser, or as Formatting Objects which can be used to generate PDF files. XSL allows you to implement look-and-feel and formatting without any modification to the XML source.

Working with XML in HyperContent

XML DTDs

DTDs define the possible structure of an XML file. DTDs are normally referred to within the prolog of an XML file in a DOCTYPE declaration. In HyperContent, this declaration may point to the URL of a DTD on the network, a URL which is mapped to a local resource by an XML catalog, or to the path of a file in the same repository as the XML file. HyperContent has a built-in XML editor that allows users to author XML files in accordance with the grammars you have specified in your DTDs, directly from their web browser. HyperContent also provides a framework for plugging in custom web-authoring interfaces to provide more tailored tools for managing files according to specific XML grammars. For an introduction to DTDs, visit w3schools.

XHTML

HyperContent's built-in XML editor also supports WYSIWYG XHTML editing. This is automatically enabled for any XML element named "html". You can either specify "html" as the root element of a document, or as a child element so that you can store XHTML along with custom structured data. HyperContent will support any number of "html" elements in a document.

XML Catalog

Oasis, an Internet standards body, has defined a standard format for a file which can be used to map URLs of Internet resources to other URLs or local copies of those resources. This is useful to decrease load time of DTDs, to protect against network outages, to allow for offline development, or to quietly "convert" XML files conforming to a certain DTD to an updated or modified version of the DTD, without removing the original. HyperContent provides an XML catalog file called "Catalog.xml" in the hypercontent properties directory. Please visit the OASIS page for more information.

Managing structure across many XML files

As a text file format, XML holds out the promise of being legible by both computers and humans. However, in dealing with large amounts of data it becomes challenging to fulfill this promise: to maintain human legibility, it is of significant value to break the data up into smaller files, while for machine processing one giant file is required to maintain coherent data structures. HyperContent makes it possible to achieve both goals by allowing you to define structure across multiple XML files, which can be managed individually by humans but processed together by the computer.

HyperContent allows you to use path patterns to identify particular data structures according to where the files are stored in the filesystem, so that you can use XML Includes to bring related data together at build time. In your XSL transformations you can work with included data based on its path pattern in order to apply common formatting, just as you would if each file were an element in one giant XML file.

There are two elements that can be used in a HyperContent project definition to define XML content:

< ***xml-doc*** >

An xml-doc tag is used to specify a single XML file that must exist in the project repository. Common uses for this are specifying a home page, XSL files, and configuration files. An xml-doc tag has three required attributes:

definition =

The location of the DTD which specifies the grammar this file conforms to. This will be used in constructing the DOCTYPE portion of the file's prolog. This can be a URL, the path of a DTD in the project repository (e.g. '/dtds/common.dtd'), or blank if this will be an xsl file. If you want to specify a document to hold a custom dictionary for the project, to be used by the spell checker, you should specify the URL http://hypercontent.sourceforge.net/dtd/dictionary.dtd.

path =

The path of this file in the project repository. e.g.

/index.xml

/xsl/common.xsl

/dictionary.xml

root =

The name of the root element of this document. e.g.

xsl:stylesheet

myelement

html

< ***xml-doctype*** >

The xml-doctype tag is used to describe a class of XML documents: it acts as a template for 0 or more XML files that share the same structure and function within the site.

definition = *(see above)*
path =

The path of an xml-doctype differs from that of an xml-doc in that it it describes a possible rather than literal location. Possibilities are expressed using wildcards: e.g. '/*/index.xml' indicates the index page of directory beneath the repository root; '/*/*' indicates a file one level down in the repository. Wildcards are replaced with actual names by authors at the time they create a new instance of this doctype. For a more complete explanation and examples, see the chapter on path patterns.

root = *(see above)*
label =

The label of a doctype should be a short, descriptive text describing a file which is an instance of this doctype. For example, if you were specifying a doctype for a faculty member page, you could label it "Faculty" or "Faculty Member Biography".

template =

This allows you to specify the path of a file in the repository which will act as a template for new instances of this doctype. When an author creates a new instance of this doctype, they will be provided with a copy of the data and metadata from the template file. If you do not specify a template, the new file will consist of an empty root element which can be populated from scratch. You may want to specify the template file as an xml-doc, to guarantee that it can not be removed from the repository.

Forms of Output

Both xml-docs and xml-doctypes can be configured with 0 or more forms of output. For example, you might configure a file to be output as standard HTML, accessible HTML, RSS and PDF. Other files may not have any forms of output, such as XSL and configuration files - these are intended to be used in the maintenance and building of the site, rather than published.

< *output* >

The output tag has two required attributes

content-type =

This attribute is used to specify the content type of the output file; this will be used to determine the appropriate file extension, and may trigger specific behavior in transformation or filters. For example, specifying a content-type of "application/pdf" triggers the XSL transformation filter to treat its output as Formatting Objects and to render these into a PDF file. "text/html" is the content type for HTML files. Note that you can only have one output form of a given content-type per base directory.

basedir =

By default, the base directory of a form of output is set to "/", which indicates that the output file will live at the same path in the output filesystem as in the repository, with only the file extension changing according to the content type. If you need to have several forms of output with the same content type, for example standard and accessible HTML, you could specify a basedir of "/" for the standard form and "/acc/" for the accessible form. The accessible version of the home page would then be output to "/acc/index.html".

The output element can be configured with 0 or more include elements, 0 or more filter elements and 0 or 1 transform elements. Follow the links to the appropriate chapters for more information.

Overriding Default Editors

By default, XML files will be configured to use the default editors as configured in the Content Types file. You can override the defaults for an xml-doc or xml-doctype by adding editor elements inside it. You must specify all the editors that you want to be used; once you add an editor element, the defaults are ignored. The editors appear in the order you specify them: thus, the first editor element will be the opening screen for an author who chooses to edit a file corresponding to this xml-doc or xml-doctype.
< *editor* >

The editor element has a single attribute:
key = *(see the chapter on Editors)*

RDF: Extensible Metadata

About metadata and RDF

Metadata is information about a resource. In the context of HyperContent, that resource can be any file that lives in a repository - whether it is an XML file, an image file, a Microsoft Word document, or any other kind of file. The most common use of metadata is to indicate a title, description and keywords about a resource, but robust metadata support requires a framework that can be extended to store any type of specialized or custom metadata.

RDF, short for Resource Description Framework, is a metadata framework standard issued by the World Wide Web Consortium (w3c), the same group responsible for HTML, XML, HTTP, CSS, DOM and many other standards which define the interoperability of web tools. RDF, as a way of presenting structured information, bears some abstract resemblance to XML, and can in fact be represented in XML. At its core, however, RDF uses a directed graph model which, while lacking the greater flexibility of XML, makes it possible to make metadata declarations and queries on the basis of simple statements.

An RDF statement has three components; a resource, a property, and a value. For example, the RDF for the file which is used to generate this page contains the statement "The resource located at '/docs/manual/develop/rdf.xml' has a property 'title' with the value 'RDF: Extensible Metadata'". In this case, the value is a simple string; it is also possible that the value of a statement might be another resource, which can also have properties. For example, the file which represents the author of this document has two statements which combined would read "The resource located at '/contributors/alex.xml' has a property 'VCard' whose value is a resource that has a property 'Full Name' with the value 'Alex Vigdor'". This chaining of resources as the values of properties of other resources allows for an XML-like nesting of metadata information.

A concept borrowed from XML that is critical to RDF is that of namespaces; a namespace represents the origin and specific intent of a set of names. These names might be the names of XML elements or RDF properties; the namespace is used to resolve ambiguity that can result, for example, if two different metadata standards both use a property with the same name, but with a different intent or structure. A namespace is identified by a URI, which is mapped to a prefix for readability. For example, the 'title' property used in HyperContent is in a namespace declared by the Dublin Core Metadata Initiative version 1.0, which uses the namespace 'http://purl.org/dc/elements/1.1/'. If that URI is mapped to the prefix 'dc', the Dublin Core title property is notated as 'dc:title'.

RDF in XML

The W3C has issued a standard for representing RDF in XML, so that XML can be used as the language for transmitting RDF information without losing the critical semantics of RDF. This standard is fairly flexible in terms of the XML structure, but this flexibility threatens to make RDF/XML very difficult to use with XPath, the query language used in XSL. HyperContent has implemented an XML representation of RDF that is compliant with the standard, but adopts a more rigid subset of the spec which is especially tailored for querying with XPath. Specifically, the resource -> property -> value relationship is represented in nested XML tags, where a property is represented as a tag with the namespace and name of the property; if the value is a resource, its properties are represented as child tags. If the value is data, it is represented as a child text node of the property tag. This XML representation is used both for metadata includes, and for the .rdf files that represent metadata when a zip archive of content is downloaded from the Content Manager.

Here's what the RDF for this file looks like serialized in XML. Note that the root element **rdf:RDF** is the container for a collection of statements, and provides the mapping of namespaces to prefixes. Four namespaces are declared here: the namespace for rdf, the 'fs' namespace for the filesystem, the 'dc' namespace for Dublin Core, and the 'cms' namespace for HyperContent. The **fs:File** tag represents the file resource, whose properties are child tags.

```
<rdf:RDF
        xmlns:fs  = "http://www.ais.columbia.edu/sws/xmlns/cufs#"
        xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc  = "http://purl.org/dc/elements/1.0/"
        xmlns:cms = "http://www.ais.columbia.edu/sws/xmlns/cucms#"
    >
    <fs:File rdf:about = "/docs/manual/develop/rdf.xml">

        <cms:comment>added sample of RDF serialized in XML</cms:comment>

        <cms:editor>Alex Vigdor</cms:editor>

        <dc:title>RDF: Extensible Metadata</dc:title>

        <dc:format>text/xml</dc:format>

        <dc:date>2003-11-06</dc:date>

        <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>

        <dc:identifier>/docs/manual/develop/rdf.xml</dc:identifier>

        <dc:creator>Alex Vigdor</dc:creator>

    </fs:File>
</rdf:RDF>
```

RDF Grammars

The next two chapters offer specific details about two RDF metadata grammars supported by custom editors in HyperContent, Dublin Core and VCard. Here we list the properties defined by two core grammars in HyperContent.

Filesystem grammar

xmlns:fs="http://www.ais.columbia.edu/sws/xmlns/cufs#"

The filesystem code which provides the repository layer for HyperContent has its own namespace; this namespace has a few properties which are used specifically for search indexing but do not appear in the RDF model generated by the filesystem. The principal property of interest for site developers is **fs:File**; the resource which describes a file in the repository has the property 'rdf:type' with the value 'fs:File'. When serialized in XML, this is reflected as an element 'fs:File' which represents the file resource; all the file's metadata properties are represented as children of this element. Thus in XPath a file's metadata properties are always located with a string like "rdf:RDF/fs:File/{ns:property}". See the example above for help visualizing the structure.

HyperContent grammar

xmlns:cms="http://www.ais.columbia.edu/sws/xmlns/cucms#"

HyperContent supports a number of built-in metadata properties which may be applied to a resource of type **fs:File**:

cms:comment

Text entered by the person who edited this file revision at the time they saved it.

cms:editor

The name of the person who saved this revision of the file.

cms:hidden

When present with a value of "true", indicates that this file should not be built or processed as an include.

cms:replacement

The value of this property specifies the path of a file or a URI for a resource from which data and metadata should be retrieved at runtime, to replace the contents of this file. Can be useful for creating a local placeholder for a network resource, or mirroring the contents of a file at different locations in the output filesystem.

cms:includes

HyperContent support ad-hoc, user-specified file includes. These may be used to make connections between two files, such as a course file and a faculty file to indicate the faculty member teaches that course. The value of the cms:includes property is a resource of type *rdf:Bag*. rdf:Bag is a special collection type of resource in RDF, which will contain one or more *rdf:li* resources. Each rdf:li resource, in turn, will have a property of type **cms:Include**, which may have the following properties:

cms:data

a value of yes indicates the data of the specified resource should be included.

cms:metadata

a value of yes indicates the metadata of the specified resource should be included.

cms:location

specifies the path in the repository or URI of the resource to be included.

cms:Thumbnail

The value of the cms:Thumbnail property is a resource which has the following properties.

cms:width

Width of the thumbnail image.

cms:height

Height of the thumbnail image.

cms:format

The mime type representing the thumbnail image format

cms:data

The binary thumbnail image data encoded in Base64

Dublin Core Metadata

The Dublin Core Metadata Initiative has established a common set of 15 metadata properties which apply to resources of all types across many areas of interest. The "Dublin Core", as the DCMI Metadata Terms are often referred to, encapsulates all of the metadata that many applications require. DCMI also has a standard binding to RDF, so using Dublin Core in HyperContent provides a way to store metadata that is completely standards based and widely understood by people and machines alike.

HyperContent has a built in editor for Dublin Core metadata; here is a description of the fifteen elements taken from the original publication, with some clarifications made in descriptions of version 1.1, and annotations with usage information specific to HyperContent.

Dublin Core 1.0 Namespace:

dc="*http://purl.org/dc/elements/1.0/*"

Title

dc:title
The name given to the resource, usually by the Creator or Publisher.

HyperContent displays the Title *of a file in the Content Manager when the file is selected, or appears in a list such as search results. It is recommended to use* Title *as the text of links to a file, and in the title tag of HTML output.*

Author or Creator

dc:creator
The person or organization primarily responsible for creating the intellectual content of the resource. For example, authors in the case of written documents, artists, photographers, or illustrators in the case of visual resources.

HyperContent automatically sets the active users name as Creator *when a new file is created*

Subject and Keywords

dc:subject
The topic of the resource. Typically, subject will be expressed as keywords or phrases that describe the subject or content of the resource. The use of controlled vocabularies and formal classification schemas is encouraged.

Description

dc:description
A textual description of the content of the resource, including abstracts in the case of document-like objects or content descriptions in the case of visual resources.

Publisher

dc:publisher
The entity responsible for making the resource available in its present form, such as a publishing house, a university department, or a corporate entity.

Other Contributor

dc:contributor
A person or organization not specified in a Creator element who has made significant intellectual contributions to the resource but whose contribution is secondary to any person or organization specified in a Creator element (for example, editor, transcriber, and illustrator).

Date

dc:date
A date associated with the creation or availability of the resource.

Recommended best practice is defined in a profile of ISO 8601 (http://www.w3.org/TR/NOTE-datetime) that includes (among others) dates of the forms YYYY and YYYY-MM-DD. In this scheme, the date 1994-11-05 corresponds to November 5, 1994.

> *HyperContent sets* Date *to current the date when a file is first created.*

Resource Type

dc:type

Type includes terms describing general categories, functions, genres, or aggregation levels for content. Recommended best practice is to select a value from a controlled vocabulary (for example, the DCMI Type Vocabulary). To describe the physical or digital manifestation of the resource, use the FORMAT element.

> *HyperContent will set the* Type *of a file when it is created to DCMI Image, Text, or Audio according to the file's MIME type. If none of those is appropriate, it is left blank. The user can select the type from a dropdown list of all the DCMI types. This list can be controlled in the properties file dcmi-types.xml.*

Format

dc:format
The data format and, optionally, dimensions (e.g., size, duration) of the resource. The format is used to identify the software and possibly hardware that might be needed to display or operate the resource. Recommended best practice is to select a value from a controlled vocabulary (for example, the list of Internet Media Types defining computer media formats).

> *HyperContent sets the* Format *of a file to it's MIME type when it is created.*

Resource Identifier

dc:identifier

A string or number used to uniquely identify the resource. Examples for networked resources include URLs and URNs (when implemented). Other globally-unique identifiers, such as International Standard Book Numbers (ISBN) or other formal names would also be candidates for this element.

> *HyperContent sets the* Identifier *of a file to its path in the repository at the time it is created.*

Source

dc:source
Information about a second resource from which the present resource is derived. While it is generally recommended that elements contain information about the present resource only, this element may contain metadata for the second resource when it is considered important for discovery of the present resource.

Language

dc:language
Recommended best practice is to use RFC 3066 which, in conjunction with ISO639, defines two- and three-letter primary language tags with optional subtags. Examples include "en" or "eng" for English, "akk" for Akkadian", and "en-GB" for English used in the United Kingdom.

> *HyperContent's built-in Dublin Core editor allows the user to select from a list of all languages represented by two letter language codes; this list can be edited in the properties file language-codes.xml. The* Language, *if set, will be used to find the best corresponding dictionary for use in the spell checker.*

Relation

dc:relation
An identifier of a second resource and its relationship to the present resource. This element is used to express linkages among related resources. Recommended best practice is to identify the referenced resource by means of a string or number conforming to a formal identification system

Coverage

dc:coverage
Typically, Coverage will include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names) and to use, where appropriate, named places or time periods in preference to numeric identifiers such as sets of coordinates or date ranges.

Rights Management

dc:rights
A rights management statement, an identifier that links to a rights management statement, or an identifier that links to a service providing information about rights management for the resource.

VCard Metadata

Contact Information: HyperContent, VCard and LDAP

As early versions of HyperContent grew to host a number of large web sites, the most common type of structured information required across all the sites turned out to be contact information, used in constructing biographical pages for faculty and students, as well as departmental directories. The first sites stored this information in XML, but a common request was for a more user friendly way to enter contact information, with the ability to retrieve as much information as possible from LDAP.

The most pervasive Internet standard related to contact information is VCard, a standard published in 1998 by the Internet Society and widely supported by email and Personal Information Management software. VCard 3.0 supports not only basic contact information (name, phone numbers, email address, postal addresses, etc), but provides support for binary data such as a photograph.

By storing VCard information as metadata, HyperContent makes it possible to associate this standard contact information with any type of file, for example an XML file containing a biography of the person. The W3C has posted a note regarding the representation of VCard data in RDF, but as the responses on the W3C site point out, the proposal is fundamentally flawed in ways that have not been addressed. The effort has therefore not produced a usable standard for the representation of VCard in RDF, so HyperContent implements a workable VCard RDF grammar in its own namespace. This is not of concern for interoperability, as HyperContent can take its RDF-encoded VCard data and publish it in the standard plain text VCard format described in the original specification, and also allows content authors to import standard VCard files from their email or PIM software directly into the VCard editor. Round trips between standard VCard encoding and HyperContent's RDF VCard encoding are lossless, even with custom X-object and X-parameters or other standard VCard elements not displayed in the VCard editor.

LDAP integration allows a user to populate standard VCard fields in the VCard editor with the results of an LDAP query, saving time and reducing the chance of errors. The configuration file LDAPtoVCARD.xml allows you to specify LDAP connection information, and has a VCard RDF template in XML which can be used to associate LDAP fields with particular VCard properties. Fields from the LDAP result are referenced by name in curly braces (e.g. {givenname}), and other values can be hardcoded (e.g. institution name).

It is important to understand the structure of HyperContent's RDF representation of

VCard in order to access VCard data in XSL transformations. For information about the values and use of specific properties, refer to the VCard RFC.

HyperContent VCard 3.0 Namespace

vcard="http://www.ais.columbia.edu/sws/xmlns/vcard3.0#"
vcard:VCARD

> The value of the vcard:VCARD property is the resource to which all other vcard properties are applied. In XPath, it will always appear between the fs:File element and the desired VCard attribute: e.g. 'rdf:RDF/fs:File/vcard:VCARD/vcard:FN/vcard:text'.

All of the capitalized properties listed below have values which are resources, which may all have common VCard properties assigned to them (listed at the end), and unless otherwise specified have a property **vcard:text** which contains the actual string data.
vcard:FN

vcard:ADR

> The value of the vcard:ADR property is a resource which may have the following properties applied to it (the values of these properties are strings containing the relevant data):
>
> **vcard:poBox**
> **vcard:extendedAddress**
>
> **vcard:streetAddress**
>
> **vcard:locality**
>
> **vcard:region**
>
> **vcard:postalCode**
>
> **vcard:countryName**

vcard:N

> The value of the vcard:N property is a resource which may have the following properties applied to it (the values of these properties are strings containing the relevant data):
>
> **vcard:familyName**
> **vcard:givenName**
>
> **vcard:additionalName**

vcard:honorificPrefix

vcard:honorificSuffix

vcard:NICKNAME

Nickname is a multivalued attribute; the values are kept in an rdf:Bag. The nodelist of nicknames can be retrieved in XPath as 'rdf:RDF/fs:File/vcard:VCARD/vcard:NICKNAME/rdf:Bag/rdf:li'.

vcard:BDAY

vcard:BDAY may have either a **vcard:text** property or a **vcard:Date** and optional **vcard:Time** property - see below for more details.

vcard:PHOTO

A photo resource may either have a **vcard:uri** property whose value is the network location of the graphic, or a **vcard:binary** property whose value is the base64 encoded image data. Note that the actual base64 data is not passed through to XSL transformations for the sake of efficiency, but testing for the presence of 'rdf:RDF/fs:File/vcard:VCARD/vcard:PHOTO/vcard:binary' will properly reflect whether there is binary image data in the source file. Also note that the common vcard property **vcard:type** has the corresponding mime image subtype as a value (HyperContent uses jpeg by default).

vcard:LABEL

vcard:TEL

The string data for vcard:TEL is the value of the **vcard:phoneNumber** property.

vcard:EMAIL

vcard:MAILER

vcard:TZ

The data for vcard:TZ is specified with either the **vcard:text** or **vcard:utcOffset** property as appropriate.

vcard:GEO

vcard:GEO has two properties which express the relevant coordinates: **vcard:longitude** and **vcard:latitude**.

vcard:TITLE
vcard:ROLE
vcard:LOGO

Same possible structure as vcard:PHOTO, see above.

vcard:AGENT

vcard:Agent has one of three properties, **vcard:uri** pointing to a reference on the network, **vcard:text** for a plain text value, or **vcard:VCARD** referencing a resource which contains VCard properties for the agent.

vcard:ORG

vcard:ORG supports a vector of values, used to express a hierarchy. The value is an rdf:Seq resource, whose rdf:li properties express the levels of the hierarchy from the top down.

vcard:CATEGORIES

Categories is a multivalued attribute; the values are kept in an rdf:Bag.

vcard:NOTE

vcard:PRODID

vcard:REV

vcard:REV may have either a **vcard:text** property or a **vcard:Date** and optional **vcard:Time** property - see below for more information.

vcard:SORT-STRING

vcard:SOUND

vcard:SOUND has the same possible structure as vcard:PHOTO (see above), except the vcard:type property is used to indicate the corresponding MIME audio subtype.

vcard:UID

vcard:URL

The address of a vcard:URL is the value of the **vcard:uri** property.

vcard:CLASS

vcard:KEY

vcard:KEY has either a **vcard:text** property or a **vcard:binary** property (with base64 encoded binary data).

vcard:X-OBJECT

Two properties apply to vcard:X-OBJECT; **vcard:xname** indicates the object's name and **vcard:text** indicates the object's value.

Date and Time

There are two special resources which model Date and Time, that may be applied as properties to either **vcard:BDAY** or **vcard:REV**

vcard:Date

vcard:Date has three properties:

vcard:year
vcard:month

vcard:day

vcard:Time

vcard:Time has three properties:

vcard:hour
vcard:minute

vcard:second

Common VCard properties

The following properties may apply to any of the top level (all-caps) VCard Resources:

vcard:type

vcard:type is used in different ways depending on the resource it is applied to: for resources with binary values, type may indicate the corresponding MIME subtype. For email, phone and address values, type provides information about the information such as whether it is preferred, whether it represents a fax or mobile phone,. etc. See the spec for full details.

vcard:value

vcard:value may be used to indicate the format of the resource data (possible values include 'uri','text','phoneNumber', etc.) See RFC 2425.

vcard:language

See RFC 2425.

vcard:encoding

See RFC 2425.

vcard:X-parameter

Two properties apply to vcard:X-parameter; **vcard:xname** indicates the parameter's name and **vcard:text** indicates its value.

XML Includes

XML Includes is a special XML aggregation filter which makes it possible to reference XML and metadata from multiple files, enabling XSL transformations to automatically generate links and navigation sections that tie a web site together.

Defining XML Includes

Any number of XML Includes can be defined per form of output of an xml-doc or xml-doctype in a HyperContent project definition. Each set of includes is configured in an include tag:
< *include* >

source =

A path pattern indicating the location of the files to be included. This may include the path of a file in the repository, a URL, or a pattern with wildcards; if the wildcard pattern does not begin with a '/', it is resolved at build time from the directory of the source file being built. See the chapter on path patterns for more information.

data = *(yes | no)*

Indicates whether the actual XML content of an included file should be copied into the output XML. This is useful when you want to reuse portions of the content, for example displaying the first 300 characters of text. Defaults to "yes", but should be set to "no" if the data is not needed, to improve build performance. If the file is not an xml content type, data will not be included.

metadata = *(yes | no)*

Indicates whether the metadata of the included file should be copied into the output XML as RDF/XML. This is useful, for example, when you want to create a link to an included file and use its title in the text of the link. Defaults to "no".

The result of XML Includes

At build time the include filters are processed first, before any XSL transformation or other filters that may be specified. The result is one XML document which contains both the source file and its metadata expressed as RDF/XML, and all of the files in the repository which match one of the configured include source patterns. There are three special elements declared in the HyperContent namespace which create the structure around the source and included data and metadata:

xmlns:cms="http://www.ais.columbia.edu/sws/xmlns/cucms#"

< *cms:wrapper* >

This is the root element of the XML document generated by the XML Includes filter. It has one **cms:source** child element representing the XML document which was passed into the filter, and 0 or more **cms:include** elements representing each included file. The **cms:source** element has two child tags, an **rdf:RDF** tag containing the source file metadata in RDF/XML, and the root element of the source file with all the source XML data beneath it. A **cms:include** element may or may not have the rdf:RDF and root element as children, depending how the include was configured. Both **cms:source** and **cms:include** have the following attributes:

directory =

The directory in which the represented file lives in the repository.

path =

The path of the represented file in the repository.

filename =

The name of the represented file in the repository (the portion of the path after the directory).

pattern =

The pattern is taken from the path attribute of the xml-doc, xml-doctype or resource-directory tag used to configure the represented file's type in the project definition. You can easily query specific page types in XPath using this attribute.

basename =

The name of this file without an extension. This can be useful for constructing links to alternate forms of output for the represented file by appending the appropriate extension.

type =

The MIME type of the represented file.

XPath examples for working with included data

To illustrate the value of XML Includes, here are a few example XPath queries that could be used in XSL transformations to leverage included data.

```
/cms:wrapper/cms:include[@pattern='/faculty/*']/child::*[not(contains(
```

Retrieves the root element of all includes which were defined as faculty pages

```
/cms:wrapper/cms:include[starts-with(@directory,/cms:wrapper/cms:sourc
```

Retrieves all included files in or beneath the same directory as the source file

```
/cms:wrapper/cms:source/rdf:RDF/fs:File/dc:title
```

The title of the source document

XSL Transformations

XSLT is a language for transforming XML documents into other XML documents; it is intended to serve as a layer between structured data and presentation, allowing you to define rules which determine how structured data is rendered into a formatting language such as (X)HTML or XSL-FO (Formatting Objects, used to generate PDF files). XSLT uses a query language called XPath to access specific data from XML files. Background information with links to references for XSLT, XSL-FO and XPath is available at the W3C XSL site.

Defining XSL transformations

Any form of output of an xml-doc or xml-doctype in a HyperContent project definition can be configured with a transform tag which sets up an XSL transformation for that content. The content type of the output may influence the transformation filter; if "application/pdf" is selected, the XSL filter will automatically engage the Formatting Objects Processor to turn the XSL-FO output of the XSLT into a PDF.
< *transform* >

source =

The path in the repository or URL of the XSL file which is used to perform the transformation.

The transform tag maybe configured with child tags to pass additional information to the XSLT as parameters:
< *with-param* >

This tag allows you to pass arbitrary parameters to your XSLT. This can be used to pass contextual information to the XSLT, enabling context-sensitive results.
name =

The name of the parameter.

value =

The value of the parameter.

< with-baseurl >

You will usually want to create links in XHTML output to other pages in the site; the baseurl is determined for each output page at build time as a relative path prefix to the root of the site output. You can prepend the baseurl to any full repository path to create a link from the output page to the output of an other repository file. For example, you might construct a link in the context of processing an XML Include to the HTML output of that file: < a href = "{concat($baseurl,@directory,@basename,'.html')}"/>

as =

The name of the parameter expected in the XSLT which should contain the baseurl value. Defaults to "baseurl".

< *with-filedate* >

In the case you want to display a last modified time on an output page, you can use this tag to pass the formatted date string as a parameter to your XSLT.

as =

The name of the parameter expected in the XSLT which should contain the filedate value. Defaults to "filedate".

Referencing Repository Content within XSLT

XSLT provides three mechanisms for referencing external data; you can set up inheritance of templates from other XSLT files using an **xsl:import** or **xsl:include** command, and you can pull in data from XML documents besides the one being processed using the **document()** function. HyperContent makes it simple to use these commands to access other files in your project repository; simply use the path of the file in the repository you want to access, and HyperContent will provide it. You can also reference content using URLs, and any rules set up in your XML catalog will be followed in retrieving the content.

Filters

HyperContent provides a filtering architecture to allow unlimited possibilities in the manipulation of XML content. This architecture allows you to implement custom filters using the IPublicationFilter interface, and declare the filters in the Filters.xml properties file. Filters can be configured for any form of output of an xml-doc or xml-doctype in a HyperContent project definition using a filter tag.

< *filter* >

key =

The key of the filter as configured in Filters.xml.

A filter tag can also have any number of with-param child elements allowing you to pass arbitrary parameters to the filter via its setParameters() method.

< *with-param* >

name=

The name of the parameter

value=

The value of the parameter

Built in Filters

HyperContent has two filters built in which allow you to extract data from XML files with VCard metadata:

vcard_photo

Used to extract the JPEG photo from VCard metadata. Use this filter if you want to output the image from a VCard as a separate file.

vcard_vcf

Use this filter to output a standard .vcf vcard file, which is compatible with most major email clients and personal information management software. This filter recognizes two parameters:

fold = (true | false)

> indicates whether the VCard should have long lines folded to the standard maximum width of 75 characters per line. Defaults to true.

photo = (true | false)

> indicates whether the photo should be included as standard base64 encoded binary data. Some, though not all clients will import this photo with the vcf file, and it will significantly increase the size of the file.

An additional filter is provided which allows for the resolution of navigation templates at build time. See the navigation chapter for more information.

Non-XML Content

< ***resource-directory*** >

HyperContent can be set up to work with any kind of file that has a MIME type - see thee chapter on Content Types for global configuration information. At the project level, any kind of file that is not XML can be configured in a resource-directory tag.

label =

The label of a resource directory should be a short, descriptive text describing a file which would go here. For example, if you were specifying a resource-directory for news images, you could label it "News Image".

path =

The path of a resource-directory describes a possible rather than literal location. Possibilities are expressed using wildcards: e.g. '/*/portrait.jpg indicates a jpeg file in a directory beneath the repository root; '/*/*' indicates a file one level down in the repository. Wildcards are replaced with actual names by authors at the time they create a new file matching this pattern. The file extension will be based on the content type of the file created. For a more complete explanation and examples, see the chapter on path patterns.

publish = *(true | false)*

Indicates whether the resources should be published or not. For example, you would normally want image files to be published, while you may not find it necessary to publish DTDs.

< ***content*** >

A resource directory is configured to accept one or more type of content with content tags.

type =

The MIME type of an allowed type of file for this directory.

Overriding Default Editors

By default, files will be configured to use the default editors for their MIME type as configured in the Content Types file. You can override the defaults for files in a resource directory by adding editor elements inside it. You must specify all the editors that you want to be used; once you add an editor element, the defaults are ignored. The editors appear in the order you specify them: thus, the first editor element will be the opening screen for an author who chooses to edit a file corresponding to this resource-directory.
< *editor* >

The editor element has a single attribute:
key = *(see the chapter on Editors)*

Editors

HyperContent provides an extensible framework for plugging in web content authoring tools, which are referred to as editors. Any number of editors can be applied to the same file, enabling a tailored mix-and-match authoring environment. For example, the Dublin Core Metadata Editor can be combined with the Image Editor on a JPEG file, with the XML editor for an XML file, or with the Upload/Download editor for a Microsoft Word document. Additional custom editors can be implemented using the IEditor interface, or the older IFileEditorServantFactory interface.

Editor configuration

All editors are configured in the Editors.xml configuration file, which maps an instance of either editor API to a key which can be used to reference that editor in project definition files and in the ContentTypes.xml global configuration file.

The global configuration sets up default associations of editors with MIME types; in a project definition, those defaults can be overridden at the project level, or at the level of an xml-doc, xml-doctype or resource-directory.

< *editor* >

Editors are configured with this tag. Multiple editors are configured with several editor tags, and the document order in which they are listed controls the order in which they appear on screen.

key =

The key of the desired editor, as mapped in the Editors.xml document

Included Editors

There are nine editors included with HyperContent, all described in the Authoring section of this manual. Here is a list of each editor, by the key used to refer to it.

xml

DTD-based XML editing with WYSIWYG XHTML enabled for any xml element with the name 'html'. Only used for xml documents.

dublincore

Supports all 15 of the Dublin Core Metadata terms, which may be applied to files of any content type.

text

Plain text editing, useful for any type of text file (e.g. DTDs, javascript, CSS)

upload

Allows upload and download of files from the desktop to the server. Useful for any file type, especially arbitrary binary files.

vcard

Allows VCard contact information to be stored in metadata. Recommended for use with XML files, which may contain additional data about the person.

buildoptions

Configures build-time options for a file, including ad-hoc XML Includes, hiding the file, and specifying a replacement for the file (similar to filesystem linking).

image

Crop and resize photos directly in the browser, with built in support for GIF, JPEG and PNG with automated conversions.

spellcheck

Spell checker. As shipped, it can check words in XML elements and in Dublin Core title, description and keywords.

navigation

Drag-and drop reordering of navigation items, plus easy addition of links and relabeling of items.

Ignored Directories

By default, HyperContent will make certain that the publication filesystem mirrors the build filesystem, which involves copying over new content and deleting unknown files. In some situations, you may want to have directories under the publication filesystem root which are not under HyperContent's control, for example to keep a copy of an old web site available. In order to prevent that content from being deleted, you can add this element your project definition file to instruct HyperContent to ignore a directory on the publication filesystem.

< ***ignore-directory*** **>**

path =

The path of the directory on the publication filesystem that should be left alone. Should begin with a '/', e.g. "/oldsite/".

Permissions

Permissions in HyperContent provide control over the basics of authorization, such as controlling read and write access to directories and files. Permissions are associated with users or groups of users. The frameworks and GUI tools used to manage groups and permissions are part of the uPortal framework. The uPortal site has some documentation available to help understand these tools:

- uPortal Groups Service developer's guide
- uPortal permissions design
- uPortal Composite Group service documentation
- uPortal Groups Manager Channel user's guide

This manual also contains a chapter on setting permissions in the Content Manager.

There are seven activities in HyperContent which can be authorized:

- **Read**: view the target repository path and preview its output
- **Write**: edit content and save changes
- **Create:** create new directories and files
- **Delete**: delete directories and files
- **Grant Permissions**: grant permissions on directories and files
- **Build:** build files
- **Publish**: publish files

Here are some useful things to keep in mind with regard to permissions in HyperContent:

- When you first install HyperContent, or when you create a new project, you have to configure permissions as a Portal Administrator (super-user), before any regular users can have access.
- New projects can only be created by users with permission to create files in the "HyperContent Project Definitions" project.
- Users are only allowed to open a project if they have read permission on the root directory of that project's repository.
- Permissions apply recursively from directories to their subdirectories, but a DENY permission on a subdirectory overrides a GRANT permission on its parent. Thus you can DENY read access to particular directories in order to hide them from some users.
- Permissions are inherited recursively by the members of a group and its subgroups, but a permission given to a subgroup overrides a permission on its parent group for users who are members of the subgroup. Thus you can DENY read access on a folder to your project group, but GRANT read access to a subgroup of project administrators.
- While HyperContent does not have formal workflow, you can grant content authors the ability to BUILD but not PUBLISH their changes, so that they must request publication from an administrator.

Navigation and Site Maps

HyperContent 1.4 introduces a new way of developing site navigation and site maps. Previously, navigation structures (such as the top and left hand navigation on this page) would be generated through a combination of XML Includes and XSL logic that sorts through the includes and decides how to order and label them. If you wanted to implement custom or ad-hoc ordering, you would have to build a custom data element into the data of XML files to store a sort key: for binary files, there was no way other than sorting on filename or title. Mixing links to external sites with automatically generated in-site navigation required maintenance of separate link files.

These concerns recurred across several of the larger and more complex sites built using HyperContent, so this new system was built to provide a better way to manage navigation. It also happens to dramatically reduce HyperContent's memory usage at build time, as it moves logic and data analysis out of XSL and into Java where it can be processed much more efficiently. There are three components to the new navigation system: a navigation templating language, a navigation editor, and a navigation resolution filter.

Navigation Templates

HyperContent provides a simple XML grammar for navigation templates. This grammar allows you to create nested navigation structures with custom labeling and sorting rules.

The URL for the navigation templates dtd is

http://hypercontent.sourceforge.net/dtd/navigation.dtd

< *nav-template* >

> A navigation template embodies a rule for producing any number of navigation items, as well as rules for how its children should be sorted. It may contain any number of child **nav-template** elements, and may contain an **overrides** element which is used to override default labeling or sorting for instances of the specified pattern.
>
> **pattern =**
>
> > A pattern describes the location of a navigation item. It may contain the path of a file in the filesystem, a pattern used to define an xml-doctype in the project definition, the URL of an external resource, or can be left blank to indicate a navigation item which is intended to be just a container or label for other items.

basedir =

The node in the filesystem from which the pattern should be resolved. A single forward slash indicates the root of the repository, while a blank value indicates that the basedir should be inherited. If this is the root template, the inherited value will be the directory in which the nav-template file is stored. Otherwise, the inherited value will reflect the resolved directory of the parent nav-template instance.

sortIndex =

This field can be used to specify an integer reflecting at what position this template should occur in the resolved list of navigation items which are its siblings.

labelWith = (title|vcard|filename|date|custom)

This defines how instances of the specified pattern should be labeled: according to the Dublin core title (metadata), vcard full name (metadata), filename, date saved, or the custom label specified in the next attribute.

customLabel

Holds the custom label value used when labelWith=custom.

maxChildren

Specifies the maximum number of navigation items that should be presented beneath this one when the templates are resolved. Child items will be sorted before the list is truncated.

sortChildren = (off|by-title|by-label|by-date|by-vcard|by-filename)
childSortOrder = (ascending|descending)

Specifies if and how sorting should be applied to navigation items which are resolved as children of an instance of this template.

suppress = (true|false)

Primarily used for override templates: indicates that items resolved to this template should NOT appear in the resolved navigation document.

target

A value passed through to the resolved navigation document, which can be used to specify target for generated links.

< overrides >

The overrides element may contain any number of nav-template children. Each of these nav-templates must represent a specific instance of the pattern of the nav-template which contains the overrides; in this way, a specific instance can be given a sortIndex of 0 to move to the top, or can have a custom label applied if, for example, the page title is too long to fit in a navigation area.

Navigation Filter

The navigation filter resolves a set of navigation templates into a navigation items document. This involves resolving instances of patterns, and processing these instances according to the templates' sorting and labeling rules. It is best to perform the resolution of navigation templates into items at build time to guarantee that any content which has been added, altered or removed since the templates were last updated is reflected appropriately in the site navigation. This type of filter is specified using the **filter** tag as a direct child of an **xml-doc** or **xml-doctype** tag; this way, the data is filtered once and can be included many times, by all the files that share navigation. The navigation filter is specified with a **filter** tag with **key="navigation"**. Other files that include a navigation file that is configured with the filter as described will receive the filtered data with a root element of navigation, as follows.

< ***navigation*** **>**

The root element of a resolved navigation items document. Contains one or more nav-item elements.

< ***nav-item*** **>**

Represents one navigation item, and may contain additional nav-item elements.

directory =

The directory which contains this item, if applicable.

location =

The filesystem path or URL of this item, if applicable.

base =

The location of this item minus the file extension and preceding period. Handy for generating links to HTML output from xml sources.

pattern =

The pattern of this item in the project definition, if applicable. Useful for grouping like items.

label =

The label of this item.

target =

The target as specified in the item's template.

Navigation Editor

The navigation editor presents a user friendly view of a navigation templates document. What the user sees is actually a hierarchical representation of the filtered result of the templates. The editor allows the user to drag and drop to re-order items within a list, quickly add one-off templates (such as links to external sites), change the label and location of items, and suppress or remove items. These changes are then converted back into templates, such that the application of the filter will yield output identical to what the user entered on screen (assuming no other files have been added, altered or removed in the repository). You can configure the navigation editor to be used with nav-template files by configuring an **editor** tag with **key="navigation"** for the relevant xml-doc or xml-doctype. It is useful to combine it with the dublincore, xml and upload editors.

Optimizations

HyperContent has been aggressively optimized for maximum performance and memory efficiency. Profiling tools were used extensively in development with a number of test sites to analyze and alleviate performance bottlenecks. Tested with the largest sites currently implemented in HyperContent, profiling indicates that the single largest performance bottleneck is processing XSL transformations. The publication report screen, described in an earlier chapter, provides metrics information detailing where the build process is spending its time. The numbers provided there can help you identify whether any particular transformations are taking a lot of time, so that you can focus your own optimization efforts where needed, and have some basis for comparing the performance implications of changes. Here are some tips for optimizing XSL processing in HyperContent:

- Use xsl keys whenever possible, to reduce the cost of xpath lookups. For example, a simple key to get a nodelist of all XML Includes conforming to a certain pattern:
  `< xsl:key name="pattern" match="/cms:wrapper/cms:include" use="@pattern"/>`
- Avoid use of the '//' axis when it is possible to define an XPath more explicitly
- For a single document, an XML Include is generally faster than calling the document() function.
- When defining XML includes, only include files you need, and only include data or metadata if you need it.
- When you need to selectively acquire the data from XML Includes based on some metadata property, it may be more efficient to include only the metadata and get the data with the document() function if you anticipate needing only a handful out of many.

Glossary

HyperContent

> *"Columbia University Content Management Suite"* - the name for the software package built at Columbia for the development and maintenance of web sites.

WYSIWYG

> *"What you see is what you get"*

XML

> eXtensible Markup Language

HyperContent terminology

build

> *Build* is a command that a qualified user can issue through the Content Manager channel in the context of a particular node in the repository of a given project; all the directories and files beneath that node will be processed according to their output rules as defined in the project definition, which may involve processing includes, XSL transformations, or other data filtering. There may be 0, 1 or many files output per repository file, depending on the configuration. The output files are stored to the build filesystem, as configured in the project definition.
>
> The build command issues a request to the request engine, and opens a request report in a separate window.

preview

publish

report

XSL related terms

FOP

> *"Formatting Objects Processor"* - an engine that processes a particular extension of XSL call "Formatting Objects", commonly used to create PDF output from an XML source. HyperContent uses an FOP provided by the Apache Foundation.

XSL

> *"eXtensible Stylesheet Language"* - a templating language used to transform XML from one form into another; you can use XSL in HyperContent to create custom XML output, HTML output, PDF output or plain text, all from the same XML source.

template

> A template is a unit of logic in an XSL Stylesheet used to process a single type of XML Element, or to perform a certain named formatting task.

Bibliography

Miloslav Nic. XSLT Reference. ZVON.

RFCs

F. Dawson, Lotus Development Corporation; T. Howes, Netscape Communications. RFC 2426; *VCard MIME Directory Profile*. The Internet Society. September 1998