Configuration

There are a numbers of ways in which DSpace may be configured and/or customized. This chapter of the documentation will discuss the configuration of the software and will also reference customizations that may be performed in the chapter following.

For ease of use, the Configuration documentation is broken into several parts:

- General Configuration - addresses general conventions used with configuring not only the dspace.cfg file, but other configuration files which use similar conventions.
- The dspace.cfg Configuration Properties File - specifies the basic `dspace.cfg` file settings
- Optional or Advanced Configuration Settings - contain other more advanced settings that are optional in the dspace.cfg configuration file.

The full table of contents follows:

1 General Configuration
    1.1 Input Conventions
    1.2 Update Reminder
2 The dspace.cfg Configuration Properties File
    2.1 The dspace.cfg file
    2.2 Main DSpace Configurations
    2.3 DSpace Database Configuration
    2.4 DSpace Email Settings
        2.4.1 Wording of E-mail Messages
    2.5 File Storage
    2.6 SRB (Storage Resource Brokerage) File Storage
    2.7 Logging Configuration
    2.8 Configuring Lucene Search Indexes
    2.9 Handle Server Configuration
    2.10 Delegation Administration : Authorization System Configuration
    2.11 Restricted Item Visibility Settings
    2.12 Proxy Settings
    2.13 Configuring Media Filters
    2.14 Crosswalk and Packager Plugin Settings
        2.14.1 Configurable MODS Dissemination Crosswalk
        2.14.2 XSLT-based Crosswalks
            2.14.2.1 Testing XSLT Crosswalks
        2.14.3 Configurable Qualified Dublin Core (QDC) dissemination crosswalk
        2.14.4 Configuring Crosswalk Plugins
        2.14.5 Configuring Packager Plugins
    2.15 Event System Configuration
    2.16 Embargo
    2.17 Checksum Checker Settings
    2.18 Item Export and Download Settings
    2.19 Subscription Emails
    2.20 Hiding Metadata
    2.21 Settings for the Submission Process
    2.22 Configuring Creative Commons License
    2.23 WEB User Interface Configurations
    2.24 Browse Index Configuration
        2.24.1 Defining the Indexes.
        2.24.2 Defining Sort Options
        2.24.3 Browse Index Normalization Rule Configuration
        2.24.4 Other Browse Options
        2.24.5 Browse Index Authority Control Configuration
    2.25 Author (Multiple metadata value) Display
    2.26 Links to Other Browse Contexts
    2.27 Recent Submissions
    2.28 Submission License Substitution Variables
    2.29 Syndication Feed (RSS) Settings
    2.30 OpenSearch Support
    2.31 Content Inline Disposition Threshold
    2.32 Multi-file HTML Document/Site Settings
    2.33 Sitemap Settings
    2.34 Authority Control Settings
    2.35 JSPUI Upload File Settings
    2.36 JSP Web Interface (JSPUI) Settings
    2.37 JSPUI Configuring Multilingual Support
        2.37.1 Setting the Default Language for the Application
        2.37.2 Supporting More Than One Language
            2.37.2.1 Changes in dspace.cfg
            2.37.2.2 Related Files
    2.38 JSPUI Item Mapper
    2.39 Display of Group Membership
    2.40 JSPUI / XMLUI SFX Server
    2.41 JSPUI Item Recommendation Setting
    2.42 Controlled Vocabulary Settings
    2.43 XMLUI Specific Configuration

2.44 DSpace SOLR Statistics Configuration
3 Optional or Advanced Configuration Settings
3.1 The Metadata Format and Bitstream Format Registries
3.1.1 Metadata Format Registries
3.1.2 Bitstream Format Registry
3.2 XPDF Filter
3.2.1 Installation Overview
3.2.2 Install XPDF Tools
3.2.3 Fetch and install jai_imageio JAR
3.2.4 Edit DSpace Configuration
3.2.5 Build and Install
3.3 Configuring Usage Instrumentation Plugins
3.3.1 The Passive Plugin
3.3.2 The Tab File Logger Plugin
3.3.3 The XML Logger Plugin

General Configuration

In the following sections you will learn about the different configuration files that you will need to edit so that you may make your DSpace installation work. Of the several configuration files which you will work with, it is the dspace.cfg file you need to learn to configure first and foremost.

In general, most of the configuration files, namely `dspace.cfg` and `xmlui.xconf` will provide a good source of information not only with configuration but also with customization (cf. Customization chapters)

Input Conventions

We will use the dspace.cfg as our example for input conventions used throughout the system. It is a basic Java properties file, where lines are either comments, starting with a '#', blank lines, or property/value pairs of the form:

```
property.name = property value
```

Some property defaults are "commented out". That is, they have a "#" preceding them, and the DSpace software ignores the config property. This may cause the feature not to be enabled, or, cause a default property to be used when the software is compiled and updated.

The property value may contain references to other configuration properties, in the form `${property.name}`. This follows the ant convention of allowing references in property files. A property may not refer to itself. Examples:

```
property.name = word1 ${other.property.name} more words
property2.name = ${dspace.dir}/rest/of/path
```

Property values can include other, previously defined values, by enclosing the property name in ${...}. For example, if your dspace.cfg contains:

```
dspace.dir = /dspace
dspace.history = ${dspace.dir}/history
```

Then the value of dspace.history property is expanded to be /dspace/history. This method is especially useful for handling commonly used file paths.

Update Reminder

Things you should know about editing `dspace.cfg` files.
It is important to remember that there are * two `dspace.cfg` files after an installation of DSpace.*

1. The "source" file that is found in `[dspace-source]/dspace/config/dspace.cfg`
2. The "runtime" file that is found in `[dspace]/config/dspace.cfg`
   The runtime file is supposed to be the **copy** of the source file, which is considered the master version. However, the DSpace server and command programs only look at the *runtime* configuration file, so when you are revising your configuration values, it is tempting to *only edit the runtime file*. **DO NOT** do this. Always make the same changes to the source version of `dspace.cfg` in addition to the runtime file. The two files should always be identical, since the source `dspace.cfg` will be the basis of your next upgrade.

To keep the two files in synchronization, you can edit your files in `[dspace-source]/dspace/config/` and then you would run the following commands:

```
cd [dspace-source]/dspace/
mvn package
cd [dspace-source]/dspace/target/dspace-<version>-build.dir
ant update_configs
```

This will copy the source `dspace.cfg` (along with other configuration files) into the runtime (`[dspace]/config`) directory.

You should remember that after editing your configuration file(s), and you are done and wish to implement the changes, you will need to:

- Run `ant -Dconfig=[dspace]/config/dspace.cfg update` if you are updating your `dspace.cfg` file and wish to see the changes appear. Follow the usual sequence with copying your webapps.
- If you edit *dspace.cfg* in *[dspace-source]/dspace/config/*, you should then run '*ant init_configs*' in the directory *[dspace-source]/dspace/target /dspace-1.5.2-build.dir* so that any changes you may have made are reflected in the configuration files of other applications, for example Apache. You may then need to restart those applications, depending on what you changed.

The `dspace.cfg` Configuration Properties File

The primary way of configuring DSpace is to edit the *dspace.cfg*. You will definitely have to do this before you can run DSpace properly. *dspace.cfg* contains basic information about a DSpace installation, including system path information, network host information, and other like items. To assist you in this endeavor, below is a place for you to write down some of the preliminary data so that you may facilitate faster configuration.

- Server IP: _____
- Host Name (Server name): _____
- dspace.url: _____
- Administrator's email: _____
- handle prefix: _____
- assetstore directory: _____
- SMTP server: _____

The *dspace.cfg* file

Below is a brief "Properties" table for the dspace.cfg file and the documented details are referenced. Please refer to those sections for the complete details of the parameter you are working with.

Property	Ref. Sect.
Basic Information	

`dspace.dir`
`dspace.hostname`
`dspace.baseUrl`
`dspace.url`
`dspace.oai.url`
`dspace.name`
	6.3.2

Database Settings	

`db.name`
`db.url`
`db.driver`
`db.username`
`db.password`
	4.2.3 or 6.3.3

Advanced Database Configuration	

`db.schema`
`db.maxconnection`
`db.maxwait`
`db.maxidle`
`db.statementpool`
`db.poolname`
	6.3.3

Email Settings	

`mail.server`
`mail.server.username`
`mail.server.password`
`mail.server.port`
`mail.from.address`
`feedback.recipient`
`mail.admin`
`alert.recipient`
`registration.notify`
`mail.charset`
`mail.allowed.referrers`
`mail.extraproperties`
`mail.server.disabled`
	6.3.4

File Storage	

`assetstore.dir`
`[assetstore.dir.1`
`assetstore.dir.2`
`assetstore.incoming]`
	6.3.5

SRB File Storage	

`srb.hosts.1`
`srb.port.1`
`srb.mcatzone.1`
`srb.mdasdomainname.1`
`srb.defaultstorageresource.1`
`srb.username.1`
`srb.password.1`
`srb.homedirectory.1`
`srb.parentdir.1`
	6.3.6

Logging Configuration	

`log.init.config`
`log.dir`
`useProxies`
	6.3.7

Search Settings	

`search.dir`
`search.max-clauses`
`search.analyzer`
`search.operator`
`search.maxfieldlengthsearch.index.n`
`search.index.1`
	6.3.8

Handle Settings	
`handle.prefix` `handle.dir`	6.3.9
Delegation Administration : Authorization System Configuration	

`core.authorization.community-admin.create-subelement`
`core.authorization.community-admin.delete-subelement`
`core.authorization.community-admin.policies`
`core.authorization.community-admin.admin-group`
`core.authorization.community-admin.collection.policies`
`core.authorization.community-admin.collection.template-item`
`core.authorization.community-admin.collection.submitters`
`core.authorization.community-admin.collection.workflows`
`core.authorization.community-admin.collection.admin-group`
`core.authorization.community-admin.item.delete`
`core.authorization.community-admin.item.withdraw`
`core.authorization.community-admin.item.reinstatiate`
`core.authorization.community-admin.item.policies`
`core.authorization.community-admin.item.create-bitstream`
`core.authorization.community-admin.item.delete-bitstream`
`core.authorization.community-admin.item-admin.cc-license`
`core.authorization.collection-admin.policies`
`core.authorization.collection-admin.template-item`
`core.authorization.collection-admin.submitters`
`core.authorization.collection-admin.workflows`
`core.authorization.collection-admin.admin-group`
`core.authorization.collection-admin.item.delete`
`core.authorization.collection-admin.item.withdraw`
`core.authorization.collection-admin.item.reinstatiate`
`core.authorization.collection-admin.item.policies`
`core.authorization.collection-admin.item.create-bitstream`
`core.authorization.collection-admin.item.delete-bitstream`
`core.authorization.collection-admin.item-admin.cc-license`
`core.authorization.item-admin.policies`
`core.authorization.item-admin.create-bitstream`
`core.authorization.item-admin.delete-bitstream`
`core.authorization.item-admin.cc-license`
	6.3.10

Restricted Item Visibility Settings	

`harvest.includerestricted.rss`
`harvest.includerestricted.oai`
`harvest.includerestricted.subscription`
	6.3.12

Proxy Settings	
`http.proxy.host` `http.proxy.port`	6.3.13
Media Filter--Format Filter Plugin Settings	

`filter.plugins`
`plugin.named.org.dspace.app.mediafilter.FormatFilter`
`filter.org.dspace.app.mediafilter.PDFFilter.inputFormats`
`filter.org.dspace.app.mediafilter.HTMLFilter.inputFormats`
`filter.org.dspace.app.mediafilter.WordFilter.inputFormats`
`filter.org.dspace.app.mediafilter.JPEGFilter.inputFormats`
`filter.org.dspace.app.mediafilter.BrandedPreviewJPEGFilter.inputFormats`
	6.3.14

Custom settings for PDFFilter	
`pdffilter.largepdfsdffilter.skiponmemoryexception`	6.3.14
Crosswalk and Packager Plugin Settings (MODS, QDC, XSLT, etc.)	

`crosswalk.mods.properties.MODS`
`crosswalk.mods.properties.mods`
`crosswalk.submission.MODS.stylesheet`
	6.3.15.1

`crosswalk.qdc.namespace.QDC.dc`
`crosswalk.qdc.namespace.QDC.dcterms`
`crosswalk.qdc.schemaLocation.QDC`
`crosswalk.qdc.properties.QDC`
`mets.submission.crosswalk.DC`
`mets.submission.preserveManifest`
`mets.submission.useCollectionTemplate`
	6.3.15

`plugin.named.org.dspace.content.crosswalk.IngestionCrosswalk`
`plugin.selfnamed.org.dspace.content.crosswalk.IngestionCrosswalk`
`plugin.named.org.dspace.content.crosswalk.DisseminationCrosswalk`
`plugin.selfnamed.org.dspace.content.crosswalk.DisseminationCrosswalk`
	6.3.15.4

`plugin.named.org.dspace.content.packager.PackageDisseminator` `plugin.named.org.dspace.content.packager.PackageIngester`	6.3.15.5
Event System Configuration	

`event.dispatcher.default.class`
`event.dispatcher.default.consumers`
`event.dispatcher.noindex.class`
`event.dispatcher.noindex.consumers`
`event.consumer.search.class`
`event.consumer.search.filters`
`event.consumer.browse.class`
`event.consumer.browse.filters`
`event.consumer.eperson.class`
`event.consumer.eperson.filters`
`event.consumer.harvester.class`
`event.consumer.harvester.filters`
`event.consumer.test.class`
`event.consumer.test.filters`
`testConsumer.verbose`
	6.3.16

Embargo Settings	

`embargo.field.terms`
`embargo.field.lift`
`embargo.field.open`
`plugin.single.org.dspace.embargo.EmbargoSetter`
`plugin.single.org.dspace.embargo.EmbargoLifter`
	6.3.17

Checksum Checker	

`plugin.single.org.dspace.checker.BitsreamDispatcher`
`checker.retention.default`
`checker.retention.CHECKSUM-MATCH`
	6.3.18

Item Export and Download Settings	

`org.dspace.app.itemexport.work.dir`
`org.dspace.app.itemexport.download.dir`
`org.dspace.app.itemexport.life.span.hours`
`org.dspace.app.itemexport.max.size`
	6.3.19

Subscription Email Option	
`eperson.subscription.onlynew`	6.3.20
Bulk (Batch) Metadata Editing	

`bulkedit.valueseparator`
`bulkedit.fieldseparator`
`bulkedit.gui-item-limit`
`bulkedit.ignore-on-export`
	6.3.21

Hide Item Metadata Fields Setting	
`metadata.hide.dc.description.provenance`	6.3.22
Submission Process	
`webui.submit.blocktheses` `webui.submit.upload.required`	6.3.23
`webui.submit.enable-cc` `webui.submit.cc-jurisdiction`	6.3.24
Settings for Thumbnail Creation	

`webui.browse.thumbnail.show`
`webui.browse.thumbnail.max.height`
`webui.browse.thumbnail.max.width`
`webui.item.thumbnail.show`
`webui.browse.thumbnail.linkbehaviour`
`thumbnail.maxwidth`
`thumbnail.maxheight`
	6.3.25

Settings for Item Preview	

`webui.preview.enabled`
`webui.preview.maxwidth`
`webui.preview.maxheight`
`webui.preview.brand`
`webui.preview.brand.abbrev`
`webui.preview.brand.height`
`webui.preview.brand.font`
`webui.preview.brank.fontpoint`
`webui.preview.dc`
	6.3.25

Settings for Content Count/Strength Information	

`webui.strengths.show` `webui.strengths.cache`	6.3.25
Browse Configuration	
`webui.browse.index.`*n*	6.3.26
`webui.itemlist.sort-option.`*n*	6.3.26
`webui.browse.medata.case-insensitive`	6.3.26.3

`webui.browse.value_columns.max`
`webui.browse.sort_columns.max`
`webui.browse.value_columns.omission_mark`
`plugin.named.org.dspace.sort.OrderFormatDelegate`
	6.3.26.4

Multiple Metadata Value Display	
`webui.browse.author-field` `webui.browse.author-limit`	6.3.27
Other Browse Contexts	
`webui.browse.link.`*n*	6.3.28
Recent Submission	

`recent.submission.sort-option`
`recent.submissions.count`
`plugin.sequence.org.dspace.plugin.CommunityHomeProcessor`
`plugin.sequence.org.dspace.plugin.CollectionHomeProcessor`
	6.3.29

Submission License Substitution Variables	
`plugin.named.org.dspace.content.license.LicenseArgumentFormatter`	6.3.30
Syndication Feed (RSS) Settings	

`webui.feed.enable`
`webui.feed.items`
`webui.feed.cache.size`
`webui.cache.age`
`webui.feed.formats`
`webui.feed.localresolve`
`webui.feed.item.title`
`webui.feed.item.date`
`webui.feed.item.description`
`webui.feed.item.author`
`webui.feed.item.dc.creator`
`webui.feed.item.dc.date`
`webui.feed.item.dc.description`
`webui.feed.logo.url`
	6.3.31

OpenSearch Settings	

`websvc.opensearch.enable`
`websvc.opensearch.uicontext`
`websvc.opensearch.svccontext`
`websvc.opensearch.autolink`
`websvc.opensearch.validity`
`websvc.opensearch.shortname`
`websvc.opensearch.longname`
`websvc.opensearch.description`
`websvc.opensearch.faviconurl`
`websvc.opensearch.samplequery`
`websvc.opensearch.tags`
`websvc.opensearch.formats`
	6.3.32

Content Inline Disposition Threshold	
`webui.content_disposition_threshold` `xmlui.content_disposition_threshold`	6.3.33
Multifile HTML Settings	
`webui.html.max-depth-guess` `xmlui.html.max-depth-guess`	6.3.34
Sitemap Settings	
`sitemap.dir` `sitemap.engineurls`	6.3.35
Authority Control Settings	

`plugin.named.org.dspace.content.authority.ChoiceAuthority`
`plugin.selfnamed.org.dspace.content.authority.ChoiceAuthority`
`lcname.url`
`sherpa.romeo.url`
`authority.minconfidence`
`xmlui.lookup.select.size`
	6.3.36

JSPUI Upload File Settings	
`upload.temp.dir` `upload.max`	6.3.37
JSP Web Interface Settings	

webui.itemdisplay.default webui.resolver.1.urn webui.resolver.1.baseurl webui.resolver.2.urn webui.resolver.2.baseurl plugin.single.org.dspace.app.webui.util.StyleSelection webui.itemdisplay.thesis.collections webui.itemdisplay.metadata-style webui.itemlist.columns webui.itemlist.widths webui.itemlist.browse.<<index name>.sort.<sort name>.columns webui.itemlist.sort<sort name>.columns webui.itemlist.browse.<browse name>.columns webui.itemlist.<sort or index name>.columns webui.itemlist.dateaccessioned.columns webui.itemlist.dateaccessioned.widths webui.itemlist.tablewidth	6.3.38
JSPUI i18n Locales / Languages	
default.locale	6.3.39
JSPUI Additional Configuration for Item Mapper	
itemmap.author.index	6.3.40
JSPUI MyDSpace Display of Group Membership	
webui.mydspace.showgroupmembership	6.3.41
JSPUI SFX Server Setting	
sfx.server.url	6.3.42
JSPUI Item Recommendation Settings	
webui.suggest.enable webui.suggest.loggedinusers.only	6.3.43
JSPUI Controlled Vocabulary Settings	
webui.controlledvocabulary.enable	6.3.44
JSPUI Session Invalidation	
webui.session.invalidate	6.3.45
XMLUI Settings (Manakin)	
xmlui.supported.locales xmlui.force.ssl xmlui.user.registration xmlui.user.editmetadata xmlui.user.assumelogin xmlui.user.logindirect xmlui.theme.allowoverrides xmlui.bundle.upload xmlui.community-list.render.full xmlui.community-list.cache xmlui.bitstream.mods xmlui.bitstream.mets xmlui.google.analytics.key xmlui.controlpanel.activity.max xmlui.controlpanel.activity.ipheader	6.3.46
SOLR Statistics Configurations	

solr.log.server solr.dbfilesolr.resolver.timeout statistics.item.authorization.adminsolr.statistics.logBots solr.statistics.query.filter.spiderIP solr.statistics.query.filter.isBot solr.spiderips.urls	6.3.49

Main DSpace Configurations

Property:	`dspace.dir`
Example Value:	`/dspace`
Informational Note:	Root directory of DSpace installation. Omit the trailing '/'. Note that if you change this, there are several other parameters you will probably want to change to match, e.g. `assetstore.dir` .
Property:	`dspace.hostname`
Example Value:	`dspace.hostname = dspace.mysu.edu`
Informational Note:	Fully qualified hostname; do not include port number.
Property:	`dspace.baseUrl`
Example Value:	http://dspacetest.myu.edu:8080
Informational Note:	Main URL at which DSpace Web UI webapp is deployed. Include any port number, but do not include the trailing '/'.
Property:	`dspace.url`
Example Value:	`dspace.url = ${dspace.baseUrl}/jspui`
Informational note	DSpace base URL. URL that determines whether JSPUI or XMLUI will be loaded by default. Include port number etc., but NOT trailing slash. Change to `/xmlui` if you wish to use the xmlui (Manakin) as the default, or remove "`/jspui`" and set webapp of your choice as the "ROOT" webapp in the servlet engine.
Property:	`dspace.oai.url`
Example Value:	`dspace.oai.url = ${dspace.baseUrl}/oai`
Informational note:	The base URL of the OAI webapp (do not include /request).
Property:	`dspace.name`
Example Value:	`dspace.name = DSpace at My University`
Informational Note:	Short and sweet site name, used throughout Web UI, e-mails and elsewhere (such as OAI protocol)

DSpace Database Configuration

Many of the database configurations are software-dependent. That is, it will be based on the choice of database software being used. Currently, DSpace properly supports PostgreSQL and Oracle.

Property:	`db.name`
Example Value:	`db.name = postgres`
Informational Note:	Both `postgres` or `oracle` are accepted parameters.
Property:	`db.url`
Example Value:	`db.url = jdbc:postgresql://localhost:5432/dspace_-services`
Informational Note:	The above value is the default value when configuring with PostgreSQL. When using Oracle, use this value: `jbdc.oracle.thin:@//host:port/dspace`
Property:	`db.username`
Example Value:	`db.username = dspace`
Informational Note:	In the installation directions, the administrator is instructed to create the user "dspace" who will own the database "dspace".
Property:	`password`
Example Value:	`password = dspace5`
Informational Note:	This is the password that was prompted during the installation process (cf. 3.2.3. Installation)
Property:	`db.schema`
Example Value:	`db.schema = vra`
Informational Note:	If your database contains multiple schemas, you can avoid problems with retrieving the definitions of duplicate objects by specifying the schema name here that is used for DSpace by uncommenting the entry. This property is optional.
Property:	`db.maxconnections`
Example Value:	`db.maxconnections = 30`
Informational Note:	Maximum number of Database connections in the connection pool
Property:	`db.maxwait`
Example Value:	`db.maxwait = 5000`
Informational Note:	Maximum time to wait before giving up if all connections in pool are busy (in milliseconds).
Property:	`db.maxidle`
Example Value:	`db.maxidle = -1`
Informational Note:	Maximum number of idle connections in pool. (*-1* = unlimited)
Property:	`db.statementpool`
Example Value:	`db.statementpool = true`
Informational Note:	Determines if prepared statement should be cached. (Default is set to true)
Property:	`db.poolname`

Example Value:	`db.poolname = dspacepool`
Informational Note:	Specify a name for the connection pool. This is useful if you have multiple applications sharing Tomcat's database connection pool. If nothing is specified, it will default to 'dspacepool'

DSpace Email Settings

The configuration of email is simple and provides a mechanism to alert the person(s) responsible for different features of the DSpace software.

Property:	`mail.server`
Example Value:	`mail.server = smtp.my.edu`
Informational Note:	The address on which your outgoing SMTP email server can be reached.
Property:	`mail.server.username`
Example Value:	`mail.server.username = myusername`
Informational Note:	SMTP mail server authentication username, if required. This property is optional.
Property:	`mail.server.password`
Example Value:	`mail.server.password = mypassword`
Informational Note:	SMTP mail server authentication password, if required. This property is optional/
Property:	`mail.server.port`
Example Value:	`mail.server.port = 25`
Informational Note:	The port on which your SMTP mail server can be reached. By default, port 25 is used. Change this setting if your SMTP mailserver is running on another port. This property is optional.
Property:	`mail.from.address`
Example Value:	`mail.from.address = dspace-noreply@myu.edu`
Informational Note:	The "From" address for email. Change the 'myu.edu' to the site's host name.
Property:	`feedback.recipient`
Example Value:	`feedback.recipient = dspace-help@myu.edu`

Informational Note:	When a user clicks on the feedback link/feature, the information will be send to the email address of choice. This configuration is currently limited to only one recipient.
Property:	`mail.admin`
Example Value:	`mail.admin = dspace-help@myu.edu`
Informational Note:	Email address of the general site administrator (Webmaster)
Property:	`alert.recipient`
Example Value:	`alert.recipient = john.doe@myu.edu`
Informational Note:	Enter the recipient for server errors and alerts. This property is optional.
Property:	`registration.notify`
Example Value:	`registration.notify = mike.smith@myu.edu`
Informational Note:	Enter the recipient that will be notified when a new user registers on DSpace. This property is optional.
Property:	`mail.charset`
Example Value:	`mail.charset = UTF-8`
Informational Note:	Set the default mail character set. This may be over-ridden by providing a line inside the email template '*charset: <encoding>*', otherwise this default is used.
Property:	`mail.allowed.referrers`
Example Value:	`mail.allowed.referrers = localhost`
Informational Note:	A comma separated list of hostnames that are allowed to refer browsers to email forms. Default behavior is to accept referrals only from *dspace.hostname*. This property is optional.
Property:	`mail.extraproperties`

Example Value:	`mail.extraproperties = mail.smtp.socketFactory.port=465, \`
	`mail.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory, \`
	`mail.smtp.socketFactory.fallback=false`

Informational Note:	If you need to pass extra settings to the Java mail library. Comma separated, equals sign between the key and the value. This property is optional.

Property:	`mail.server.disabled`
Example Value:	`mail.server.disabled = false`
Informational Note:	An option is added to disable the mailserver. By default, this property is set to '`false`'. By setting value to '`true`', DSpace will not send out emails. It will instead log the subject of the email which should have been sent. This is especially useful for development and test environments where production data is used when testing functionality. This property is optional.
Property:	`default.language`
Example Value:	`default.language = en_US`
Informational Note:	If no other language is explicitly stated in the *input-forms.xml*, the default language will be attributed to the metadata values.

Wording of E-mail Messages

Sometimes DSpace automatically sends e-mail messages to users, for example, to inform them of a new work flow task, or as a subscription e-mail alert. The wording of emails can be changed by editing the relevant file in `[dspace]/config/emails` . Each file is commented. Be careful to keep the right number 'placeholders' (e.g.*{2}*).

Note: You should replace the contact-information "`dspace-help@myu.edu or call us at xxx-555-xxxx`" with your own contact details in:
`config/emails/change_password`
`config/emails/register`

File Storage

DSpace supports two distinct options for storing your repository bitstreams (uploaded files). The files are not stored in the database in which Metadata, user information, ... are stored. An assetstore is a directory on your server, on which the bitstreams are stored and consulted afterwards. The usage of different assetstore directories is the default "technique" in DSpace. The parameters below define which assetstores are present, and which one should be used for newly incoming items. As an alternative, DSpace can also use SRB (Storage Resource Brokerage) as an alternative. See SRB File Storage for details regarding SRB.

Property:	`assetstore.dir`
Example Value:	`assetstore.dir = ${dspace.dir}/assetstore`
Informational Note:	This is Asset (bitstream) store number 0 (Zero). You need not place your assetstore under the */dspace* directory, but may want to place it on a different logical volume on the server that DSpace resides. So, you might have something like this: `assetstore.dir = /storevgm /assestore` .
Property:	`assetstore.dir.1` `assetstore.dir.2`
Example Value:	`assetstore.dir.1 = /second/assetstore` `assetstore.dir.2 = /third/assetstore`
Informational Note:	This property specifies extra asset stores like the one above, counting from one (1) upwards. This property is commented out (#) until it is needed.
Property:	`assetstore.incoming`
Example Value:	`assetstore.incoming = 1`

Informational Note:	Informational Note: Specify the number of the store to use for new bitstreams with this property. The default is 0 [zero] which corresponds to the 'assestore.dir' above. As the asset store number is stored in the item metadata (in the database), always keep the assetstore numbering consistent and don't change the asset store number in the item metadata.

Be Careful

In the examples above, you can see that your storage does not have to be under the `/dspace` directory. For the default installation it needs to reside on the same server (unless you plan to configure SRB (see below)). So, if you added storage space to your server, and it has a different logical volume/name /directory, you could have the following as an example:

```
assetstore.dir = /storevgm/assetstore
assetstore.dir.1 = /storevgm2/assetstore
assetstore.incoming = 1
```

Please Note: When adding additional storage configuration, you will then need to uncomment and declare `assestore.incoming = 1`

SRB (Storage Resource Brokerage) File Storage

An alternate to using the default storage framework is to use Storage Resource Brokerage (SRB). This can provide a different level of storage and disaster recovery. (Storage can take place on storage that is off-site.) Refer to http://www.sdsc.edu/srb/index.php/Main_Page for complete details regarding SRB.

The same framework is used to configure SRB storage. That is, the asset store number (0..n) can reference a file system directory as above or it can reference a set of SRB account parameters. But any particular asset store number can reference one or the other but not both. This way traditional and SRB storage can both be used but with different asset store numbers. The same cautions mentioned above apply to SRB asset stores as well. The particular asset store a bitstream is stored in is held in the database, so don't move bitstreams between asset stores, and do not renumber them.

Property:	`srb.hosts.1`
Example value:	`srb.hosts.1 = mysrbmcathost.myu.edu`
Property:	`srb.port.1`
Example value:	`srb.port.1 = 5544`
Property:	`srb.mcatzone.1`
Example value:	`srb.mcatzone.1 = mysrbzone`
Informational Note:	Your SRB Metadata Catalog Zone. An SRB Zone (or zone for short) is a set of SRB servers 'brokered' or administered through a single MCAT. Hence a zone consists of one or more SRB servers along with one MCAT-enabled server. Any existing SRB system (version 2.x.x and below) can be viewed as an SRB zone. For more information on zones, please check http://www.sdsc.edu/srb/index.php/Zones.
Property:	`srb.mdasdomainname.1`
Example Value:	`srb.mdasdomainname.1 = mysrbdomain`
Informational Note:	Your SRB domain. This domain should be created under the same zone, specified in srb.mcatzone. Information on domains is included here http://www.sdsc.edu/srb/index.php/Zones.
Property:	`srb.defaultstorageresource.1`
Example Value:	`srb.defaultstorageresource.1 = mydefaultsrbresource`

Informational Note:	Your default SRB Storage resource.
Property:	`srb.username.1`
Example Value:	`srb.username.1 = mysrbuser`
Informational Note:	Your SRB Username.
Property:	`srb.password.1`
Example Value:	`srb.password.1 = mysrbpassword`
Informational Note:	Your SRB Password.
Property:	`srb.homedirectory.1`
Example Value:	`srb.homedirectory.1 =` `/mysrbzone/home/ mysrbuser.mysrbdomain`
Informational Note:	Your SRB Homedirectory
Property:	`srb.parentdir.1`
Example Value:	`srb.parentdir.1 = mysrbdspaceassetstore`
Informational Note:	Several of the terms, such as mcatzone, have meaning only in the SRB context and will be familiar to SRB users. The last, `srb.paratdir.n`, can be used for additional (SRB) upper directory structure within an SRB account. This property value could be blank as well.

The 'assetstore.incoming' property is an integer that references where **new** bitstreams will be stored. The default (say the starting reference) is zero. The value will be used to identify the storage where all new bitstreams will be stored until this number is changed. This number is stored in the Bitstream table (store_number column) in the DSpace database, so older bitstreams that may have been stored when '*asset.incoming*' had a different value can be found.

In the simple case in which DSpace uses local (or mounted) storage the number can refer to different directories (or partitions). This gives DSpace some level of scalability. The number links to another set of properties 'assetstore.dir', 'assetstore.dir.1' (remember zero is default), assetstore.dir.2', etc., where the values are directories.

To support the use of SRB DSpace uses the same scheme but broaden to support:

- using SRB instead of the local file system
- using the local file system (native DSpace)
- using a mix of SRB and local file system
  in this broadened use of the 'asset.incoming' integer will refer to one of the following storage locations:

- a local file system directory (native DSpace)
- a set of SRB account parameters (host, port, zone, domain, username, password, home directory, and resource
  Should there be any conflict, like '2' referring to a local directory and to a set of SRB parameters, the program will select the local directory.

If SRB is chosen from the first install of DSpace, it is suggested that 'assetstore.dir' (no integer appended) be retained to reference a local directory (as above under File Storage) because build.xml uses this value to do a `mkdir`. In this case, 'assetstore.incoming' can be set to 1 (i.e. uncomment the line in File Storage above) and the 'assetstore.dir' will not be used.

Logging Configuration

Property:	`log.init.config`
Example Value:	`log.init.config = ${dspace.dir}/config/log4j.properties`

Informational Note:	This is where your logging configuration file is located. You may override the default log4j configuration by providing your own. Existing alternatives are:
	`log.init.config = ${dspace.dir}/config/log4j.properties`
	`log.init.config = ${dspace.dir}/config/log4j-console.properties`

Property:	`log.dir`
Example value:	`log.dir = ${dspace.dir}/log`
Informational Note:	This is where to put the logs. (This is used for initial configuration only)
Property:	`useProxies`
Example Value:	`useProxies = true`
Informational Note:	If your DSpace instance is protected by a proxy server, in order for log4j to log the correct IP address of the user rather than of the proxy, it must be configured to look for the X-Forwarded-For header. This feature can be enabled by ensuring this setting is set to *true.* This also affects IPAuthentication, and should be enabled for that to work properly if your installation uses a proxy server.

Previous releases of DSpace provided an example *${dspace.dir}/config/log4j.xml* as an alternative to *log4j.properties*. This caused some confusion and has been removed. log4j continues to support both Properties and XML forms of configuration, and you may continue (or begin) to use any form that log4j supports.

Configuring Lucene Search Indexes

Search indexes can be configured and customized easily in the *dspace.cfg* file. This allows institutions to choose which DSpace metadata fields are indexed by Lucene.

Property:	`search.dir`
Example Value:	`search.dir = ${dspace.dir}/search`
Informational Note:	Where to put the search index files

Property:	`search.max-clauses`
Example Value:	`search.max-clauses = 2048`
Informational Note:	By setting higher values of search.max-clauses will enable prefix searches to work on larger repositories.
Property:	`search.index.delay`
Example Value:	`search.index.delay = 5000`
Informational Note:	It is possible to create a 'delayed index flusher'. If a web application pushes multiple search requests (i.e. a barrage or sword deposits, or multiple quick edits in the user interface), then this will combine them into a single index update. You set the property key to the number of milliseconds to wait for an update. The example value will hold a Lucene update in a queue for up to 5 seconds. After 5 seconds all waiting updates will be written to the Lucene index.
Property:	`search.analyzer`
Example Value:	`search.analyzer = org.dspace.search.DSAnalyzer`
Informational Note:	Which Lucene Analyzer implementation to use. If this is omitted or commented out, the standard DSpace analyzer (designed for English) is used by default. This standard DSpace analyzer removes common stopwords, lowercases all words and performs stemming (removing common word endings, like "ing", "s", etc).
Property:	`search.analyzer`
Example Value:	`search.analyzer = org.dspace.search.DSNonStemmingAnalyzer`
Informational Note:	Instead of the standard DSpace Analyzer (DSAnalyzer), use an analyzer which doesn't "stem" words/terms. When using this analyzer, a search for "wellness" will always return items matching "wellness" and not "well". However, similarly a search for "experiments" will only return objects matching "experiments" and not "experiment" or "experimenting". When using this analyzer, you may still use WildCard searches like "experiment*" to match the beginning of words.
Property:	`search.analyzer`
Example Value:	`search.analyzer = org.apache.lucene.analysis.cn.ChineseAnalyzer`

Informational Note:	Instead of the standard English analyzer, the Chinese analyzer is used.
Property:	`search.operator`
Example Value:	`search.operator = OR`
Informational Note	Boolean search operator to use. The currently supported values are OR and AND. If this configuration item is missing or commented out, OR is used. AND requires all the search terms to be present. OR requires one or more search terms to be present.
Property:	`search.maxfieldlength`
Example Value:	`search.maxfieldlength = 10000`
Informational Note:	This is the maximum number of terms indexed for a single field in Lucene. The default is 10,000 words, often not enough for full-text indexing. If you change this, you will need to re-index for the change to take effect on previously added items. *-1* = unlimited (Integer.MAG_VALUE)
Property:	`search.index.`*n*
Example Value:	`search.index.1 = author:dc.contributor.*`
Informational Note	This property determines which of the metadata fields are being indexed for search. As an example, if you do not include the title field here, searching for a word in the title will not be matched with the titles of your items..

For example, the following entries appear in the default DSpace installation:

```
search.index.1 = author:dc.contributor.*
search.index.2 = author:dc.creator.*
search.index.3 = title:dc.title.*
search.index.4 = keyword:dc.subject.*
search.index.5 = abstract:dc.description.abstract
search.index.6 = author:dc.description.statementofresponsibility
search.index.7 = series:dc.relation.ispartofseries
search.index.8 = abstract:dc.description.tableofcontents
search.index.9 = mime:dc.format.mimetype
search.index.10 = sponsor:dc.description.sponsorship
search.index.11 = id:dc.identifier.*
search.index.11 = language:dc.language.iso
```

The format of each entry is `search.index.<id> = <search label> : <schema> . <metadata field>` where:

`<id>`	is an incremental number to distinguish each search index entry
`<search label>`	is the identifier for the search field this index will correspond to
`<schema>`	is the schema used. Dublin Core (DC) is the default. Others are possible.

`<metadata field>`	is the DSpace metadata field to be indexed.

In the example above, `search.index.1` and `search.index.2` and `search.index.3` are configured as the `author` search field. The `author` index is created by Lucene indexing all `dc.contributor.*`,`dc.creator.*` and `description.statementofresponsibility` metadata fields.

After changing the configuration run `/[dspace]/bin/dspace index-init` to regenerate the indexes.

While the indexes are created, this only affects the search results and has no effect on the search components of the user interface. One will need to customize the user interface to reflect the changes, for example, to add the a new search category to the Advanced Search.

In the above examples, notice the asterisk (*). The metadata field (at least for Dublin Core) is made up of the "element" and the "qualifier". The asterisk is used as the "wildcard". So, for example, `keyword.dc.subject.*` will index all subjects regardless if the term resides in a qualified field. (subject versus subject.lcsh). One could customize the search and only index LCSH (Library of Congress Subject Headings) with the following entry `keyword:dc.subject.lcsh` *instead of* `keyword:dc.subject.*`

Authority Control Note:

Although DSIndexer automatically builds a separate index for the authority keys of any index that contains authority-controlled metadata fields, the "Advanced Search" UIs does not allow direct access to it. Perhaps it will be added in the future. Fortunately, the OpenSearch API lets you submit a query directly to the Lucene search engine, and this may include the authority-controlled indexes.

Handle Server Configuration

The CNRI Handle system is a 3rd party service for maintaining persistent URL's. For a nominal fee, you can register a handle prefix for your repository. As a result, your repository items will be also available under the links http://handle.net/<<handle prefix>>/<<item id>>. As the base url of your repository might change or evolve, the persistent handle.net URL's secure the consistency of links to your repository items. For complete information regarding the Handle server, the user should consult Section 3.4.4.. The Handle Server section of Installing DSpace.

Property:	`handle.canonical.prefix`
Example Value	handle.canonical.prefix = http://hdl.handle.net/ handle.canonical.prefix = ${dspace.url}/handle/
Informational Note:	Canonical Handle URL prefix. By default, DSpace is configured to use http://hdl.handle.net/ as the canonical URL prefix when generating `dc.identifier.uri` during submission, and in the 'identifier' displayed in item record pages. If you do not subscribe to CNRI's handle service, you can change this to match the persistent URL service you use, or you can force DSpace to use your site's URL, e.g. `handle.canonical.prefix = ${dspace.url}/handle/`. Note that this will not alter `dc.identifer.uri` metadata for existing items (only for subsequent submissions).
Property:	`handle.prefix`
Example Value	`handle.prefix = 1234.56789`
Informational Note:	The default installed by DSpace is `123456789` but you will replace this upon receiving a handle from CNRI.
Property:	`handle.dir`

Example Value:	`handle.dir = ${dspace.dir}/handle-server`
Informational Note:	The default files, as shown in the Example Value is where DSpace will install the files used for the Handle Server.

For complete information regarding the Handle server, the user should consult 3.3.4. The Handle Server section of Installing DSpace.

Delegation Administration : Authorization System Configuration

(Authorization System Configuration)
It is possible to delegate the administration of Communities and Collections. This functionality eliminates the need for an Administrator Superuser account for these purposes. An EPerson that will be attributed Delegate Admin rights for a certain community or collection will also "inherit" the rights for underlying collections and items. As a result, a community admin will also be collection admin for all underlying collections. Likewise, a collection admin will also gain admin rights for all the items owned by the collection.

Authorization to execute the functions that are allowed to user with WRITE permission on an object will be attributed to be the ADMIN of the object (e.g. community/collection/admin will be always allowed to edit metadata of the object). The default will be "*true*" for all the configurations.

Community Administration: Subcommunities and Collections	
Property:	`core.authorization.community-admin.create-subelement`
Example Value:	`core.authorization.community-admin.create-subelement = true`
Informational Note:	Authorization for a delegated community administrator to create subcommunities or collections.
Property:	`core.authorization.community-admin.delete-subelement`
Example Value:	`core.authorization.community-admin.delete-subelement = true`
Informational Note:	Authorization for a delegated community administrator to delete subcommunities or collections.
Community Administration: Policies and The group of administrators	
Property:	`core.authorization.community-admin.policies`
Example Value:	`core.authorization.community-admin.policies = true`
Informational Note:	Authorization for a delegated community administrator to administrate the community policies.
Property:	`core.authorization.community-admin.admin-group`
Example Value:	`core.authorization.community-admin.admin-group = true`
Informational Note:	Authorization for a delegated community administrator to edit the group of community admins.
Community Administration: Collections in the above Community	
Property:	`core.authorization.community-admin.collection.policies`
Example Value:	`core.authorization.community-admin.collection.policies = true`
Informational Note:	Authorization for a delegated community administrator to administrate the policies for underlying collections.
Property:	`core.authorization.community-admin.collection.template-item`
Example Value:	`core.authorization.community-admin.collection.template-item = true`

Informational Note:	Authorization for a delegated community administrator to administrate the item template for underlying collections.
Property:	`core.authorization.community-admin.collection.submitters`
Example Value:	`core.authorization.community-admin.collection.submitters = true`
Informational Note:	Authorization for a delegated community administrator to administrate the group of submitters for underlying collections.
Property:	`core.authorization.community-admin.collection.workflows`
Example Value:	`core.authorization.community-admin.collection.workflows = true`
Informational Note:	Authorization for a delegated community administrator to administrate the workflows for underlying collections.
Property:	`core.authorization.community-admin.collection.admin-group`
Example Value:	`core.authorization.community-admin.collection.admin-group = true`
Informational Note:	Authorization for a delegated community administrator to administrate the group of administrators for underlying collections.
Community Administration: Items Owned by Collections in the Above Community	
Property:	`core.authorization.community-admin.item.delete`
Example Value:	`core.authorization.community-admin.item.delete = true`
Informational Note:	Authorization for a delegated community administrator to delete items in underlying collections.
Property:	`core.authorization.community-admin.item.withdraw`
Example Value:	`core.authorization.community-admin.item.withdraw = true`
Informational Note:	Authorization for a delegated community administrator to withdraw items in underlying collections.
Property:	`core.authorization.community-admin.item.reinstate`
Example Value:	`core.authorization.community-admin.item.reinstate = true`
Informational Note:	Authorization for a delegated community administrator to reinstate items in underlying collections.
Property:	`core.authorization.community-admin.item.policies`
Example Value:	`core.authorization.community-admin.item.policies = true`
Informational Note:	Authorization for a delegated community administrator to administrate item policies in underlying collections.
Community Administration: Bundles of Bitstreams, related to items owned by collections in the above Community	
Property:	`core.authorization.community-admin.item.create-bitstream`
Example Value:	`core.authorization.community-admin.item.create-bitstream = true`
Informational Note:	Authorization for a delegated community administrator to create additional bitstreams in items in underlying collections.
Property:	`core.authorization.community-admin.item.delete-bitstream`
Example Value:	`core.authorization.community-admin.item.delete-bitstream = true`
Informational Note:	Authorization for a delegated community administrator to delete bitstreams from items in underlying collections.
Property:	`core.authorization.community-admin.item.cc-license`
Example Value:	`core.authorization.community-admin.item.cc-license = true`

Informational Note:	Authorization for a delegated community administrator to administer licenses from items in underlying collections.

Community Administration:
The properties for collection administrators work similar to those of community administrators,
with respect to collection administration.
	`core.authorization.collection-admin.policies`
	`core.authorization.collection-admin.template-item`
	`core.authorization.collection-admin.submitters`
	`core.authorization.collection-admin.workflows`
	`core.authorization.collection-admin.admin-group`

Collection Administration:
Item owned by the above CollectionThe properties for collection administrators work similar to those of
community administrators,
with respect to administration of
items in underlying collections.
	`core.authorization.collection-admin.item.delete`
	`core.authorization.collection-admin.item.withdraw`
	`core.authorization.collection-admin.item.reinstatiate`
	`core.authorization.collection-admin.item.policies`

Collection Administration:
Bundles of bitstreams, related to items owned by collections in the above Community. The properties for collection administrators work similar to those of community administrators, with respect to administration of bitstreams related to items in underlying collections.
	`core.authorization.collection-admin.item.create-bitstream`
	`core.authorization.collection-admin.item.delete-bitstream`
	`core.authorization.collection-admin.item-admin.cc-license`

Item Administration. The properties for item administrators work similar to those of community and collection administrators, with respect to administration of items in underlying collections.	`core.authorization.item-admin.policies`

Item Administration:
Bundles of bitstreams, related to items owned by collections in the above Community. The properties for item administrators work similar to those of community and collection administrators,
with respect to administration of bitstreams
related to items in underlying collections.
	`core.authorization.item-admin.create-bitstream`
	`core.authorization.item-admin.delete-bitstream`
	`core.authorization.item-admin.cc-license`

Oracle users should consult *Chapter 4 Updating a DSpace Installation* regarding the necessary database changes that need to take place.

Restricted Item Visibility Settings

By default RSS feeds, OAI-PMH and subscription emails will include ALL items regardless of permissions set on them. If you wish to only expose items through these channels where the ANONYMOUS user is granted READ permission, then set the following options to false.

In large repositories, setting harvest.includerestricted.oai to false may cause performance problems as all items will need to have their authorization permissions checked, but because DSpace has not implemented resumption tokens in ListIdentifiers, ALL items will need checking whenever a ListIdentifers request is made.

Property:	`harvest.includerestricted.rss`
Example Value:	`harvest.includerestricted.rss = true`
Informational Note:	When set to 'true' (default), items that haven't got the READ permission for the ANONYMOUS user, will be included in RSS feeds anyway.
Property:	`harvest.includerestricted.oai`
Example Value:	`harvest.includerestricted.oai = true`
Informational Note:	When set to true (default), items that haven't got the READ permission for the ANONYMOUS user, will be included in OAI sets anyway.
Property:	`harvest.includerestricted.subscription`
Example Value:	`harvest.includerestricted.subscription = true`
Informational Note:	When set to true (default), items that haven't got the READ permission for the ANONYMOUS user, will be included in Subscription emails anyway.

Proxy Settings

These settings for proxy are commented out by default. Uncomment and specify both properties if proxy server is required for external http requests. Use regular host name without port number.

Property:	`http.proxy.host`
Example Value	`http.proxy.host = proxy.myu.edu`
Informational Note	Enter the host name without the port number.
Property:	`http.proxy.port`
Example Value	`http.proxy.port = 2048`
Informational Note	Enter the port number for the proxy server.

Configuring Media Filters

Media or Format Filters are classes used to generate derivative or alternative versions of content or bitstreams within DSpace. For example, the PDF Media Filter will extract textual content from PDF bitstreams, the JPEG Media Filter can create thumbnails from image bitstreams.

Media Filters are configured as Named Plugins, with each filter also having a separate configuration setting (in *dspace.cfg*) indicating which formats it can process. The default configuration is shown below.

Property:	`filter.plugins`
Example Value:	`filter.plugins = PDF Text Extractor, Html Text Extractor, \` `  Word Text Extractor, JPEG Thumbnail`

Informational Note:	Place the names of the enabled MediaFilter or FormatFilter plugins. To enable Branded Preview, comment out the previous one line and then uncomment the two lines in found in *dspace.cfg*:
	`Word Text Extractor, JPEG Thumbnail,\`
	`  Branded Preview JPEG`

Property:	`plugin.named.org.dspace.app.mediafilter.FormatFilter`

Example Value:	`plugin.named.org.dspace.app.mediafilter.FormatFilter = \`
	`  org.dspace.app.mediafilter.PDFFilter = PDF Text Extractor, \`
	`  org.dspace.app.mediafilter.HTMLFilter = HTML Text Extractor, \`
	`  org.dspace.app.mediafilter.WordFilter = Word Text Extractor, \`
	`  org.dspace.app.mediafilter.JPEGFilter = JPEG Thumbnail, \`
	`  org.dspace.app.mediafilter.BrandedPreviewJPEGFilter = Branded Preview JPEG`

Informational Note:	Assign "human-understandable" names to each filter

Property:	`filter.org.dspace.app.mediafilter.PDFFilter.inputFormats`
	`filter.org.dspace.app.mediafilter.HTMLFilter.inputFormats`
	`filter.org.dspace.app.mediafilter.WordFilter.inputFormats`
	`filter.org.dspace.app.mediafilter.JPEGFilter.inputFormats`
	`filter.org.dspace.app.mediafilter.BrandedPreviewJPEGFilter.inputFormats`

Example Value:	`filter.org.dspace.app.mediafilter.PDFFilter.inputFormats = Adobe PDF`
	`filter.org.dspace.app.mediafilter.HTMLFilter.inputFormats = HTML, Text`
	`filter.org.dspace.app.mediafilter.WordFilter.inputFormats = Microsoft Word`
	`filter.org.dspace.app.mediafilter.JPEGFilter.inputFormats = BMP, GIF, JPEG, \`
	`image/png`
	`filter.org.dspace.app.mediafilter.BrandedPreviewJPEGFilter.inputFormats = BMP, \`
	`GIF, JPEG, image/png`

Informational Note:	Configure each filter's input format(s)
Property:	`pdffilter.largepdfs`
Example Value:	`pdffilter.largepdfs = true`
Informational Note:	It this value is set for "true", all PDF extractions are written to temp files as they are indexed. This is slower, but helps to ensure that PDFBox software DSpace uses does not eat up all your memory.
Property:	`pdffilter.skiponmemoryexception`
Example Value:	`pdffilter.skiponmemoryexception = true`
Informational Note:	If this value is set for "true", PDFs which still result in an "Out of Memory" error from PDFBox are skipped over. These problematic PDFs will never be indexed until memory usage can be decreased in the PDFBox software.

Names are assigned to each filter using the `plugin.named.org.dspace.app.mediafilter.FormatFilter` field (e.g. by default the PDFilter is named "PDF Text Extractor".

Finally, the appropriate `filter.<class path>.inputFormats` defines the valid input formats which each filter can be applied. These format names **must match** the `short description` field of the Bitstream Format Registry.

You can also implement more dynamic or configurable Media/Format Filters which extend `SelfNamedPlugin` .

More Information on MediaFilters

For more information on Media/Format Filters, see the section on Transforming DSpace Content (MediaFilters).

Crosswalk and Packager Plugin Settings

The subsections below give configuration details based on the types of crosswalks and packager plugins you need to implement.

More Information on Packagers & Crosswalks

For more information on using Packagers and Crosswalks, see the section on Importing and Exporting Content via Packages.

Configurable MODS Dissemination Crosswalk

The MODS crosswalk is a self-named plugin. To configure an instance of the MODS crosswalk, add a property to the DSpace configuration starting with "c`rosswalk.mods.properties.`"; the final word of the property name becomes the plugin's name. For example, a property name `crosswalk.mods.properties.MODS` defines a crosswalk plugin named "`MODS`".

The value of this property is a path to a separate properties file containing the configuration for this crosswalk. The pathname is relative to the DSpace configuration directory, i.e. the `config` subdirectory of the DSpace install directory. Example from the `dspace.cfg` file:

Properties:	`crosswalk.mods.properties.MODS` `crosswalk.mods.properties.mods`
Example Values:	`crosswalk.mods.properties.MODS = crosswalks/mods.properties` `crosswalk.mods.properties.mods = crosswalks/mods.properties`
Informational Note:	This defines a crosswalk named MODS whose configuration comes from the file `[dspace]/config/crosswalks/mods.properties` . (In the above example, the lower-case name was added for OAI-PMH)

The MODS crosswalk properties file is a list of properties describing how DSpace metadata elements are to be turned into elements of the MODS XML output document. The property name is a concatenation of the metadata schema, element name, and optionally the qualifier. For example, the *contributor.author* element in the native Dublin Core schema would be: *dc.contributor.author*. The value of the property is a line containing two segments separated by the vertical bar ("|"_): The first part is an XML fragment which is copied into the output document. The second is an XPath expression describing where in that fragment to put the value of the metadata element. For example, in this property:

```
dc.contributor.author = <mods:name>
                                <mods:role>
                                        <mods:roleTerm type="text">author</mods:roleTerm>
                                <mods:role>
                                        <mods:namePart>%s</mods:namePart>
                        </mods:name>
```

Some of the examples include the string "%s" in the prototype XML where the text value is to be inserted, but don't pay any attention to it, it is an artifact that the crosswalk ignores. For example, given an author named *Jack Florey*, the crosswalk will insert

```
<mods:name>
   <mods:role>
     <mods:roleTerm type="text">author</mods:roleTerm>
   </mods:role>
   <mods:namePart>Jack Florey</mods:namePart>
</mods:name>
```

into the output document. Read the example configuration file for more details.

XSLT-based Crosswalks

The XSLT crosswalks use XSL stylesheet transformation (XSLT) to transform an XML-based external metadata format to or from DSpace's internal metadata. XSLT crosswalks are much more powerful and flexible than the configurable MODS and QDC crosswalks, but they demand some esoteric knowledge (XSL stylesheets). Given that, you can create all the crosswalks you need just by adding stylesheets and configuration lines, without touching any of the Java code.

The default settings in the `dspace.cfg` file for submission crosswalk:

Properties:	`crosswalk.submission.MODS.stylesheet`
Example Value:	`crosswalk.submission.MODS.stylesheet = crosswalks/mods-submission.xsl`
Informational Note:	Configuration XSLT-driven submission crosswalk for MODS

As shown above, there are three (3) parts that make up the properties "key":

```
crosswalk.submissionPluginName.stylesheet =
          1        2        3               4
```

`crosswalk` first part of the property key.
`submission` second part of the property key.
`PluginName` is the name of the plugin. The *path* value is the path to the file containing the crosswalk stylesheet (relative to `/[dspace]/config`).
Here is an example that configures a crosswalk named "LOM" using a stylesheet in `[dspace]/config/crosswalks/d-lom.xsl`:
`crosswalk.submission.LOM.stylesheet = crosswalks/d-lom.xsl`
A dissemination crosswalk can be configured by starting with the property key *crosswalk.dissemination*. Example:
`crosswalk.dissemination.PluginName.stylesheet = path`
The *PluginName* is the name of the plugin (!) . The *path* value is the path to the file containing the crosswalk stylesheet (relative to `/[dspace]/config`).

You can make two different plugin names point to the same crosswalk, by adding two configuration entries with the same path:

```
crosswalk.submission.MyFormat.stylesheet = crosswalks/myformat.xslt
     crosswalk.submission.almost_DC.stylesheet = crosswalks/myformat.xslt
```

The dissemination crosswalk must also be configured with an XML Namespace (including prefix and URI) and an XML schema for its output format. This is configured on additional properties in the DSpace configuration:

```
crosswalk.dissemination.PluginName.namespace.Prefix = namespace-URI
     crosswalk.dissemination.PluginName.schemaLocation = schemaLocation value
```

For example:

```
crosswalk.dissemination.qdc.namespace.dc = http://purl.org/dc/elements/1.1/
     crosswalk.dissemination.qdc.namespace.dcterms = http://purl.org/dc/terms/
     crosswalk.dissemination.qdc.schemalocation = http://purl.org/dc/elements/1.1/ \
     http://dublincore.org/schemas/xmls/qdc/2003/04/02/qualifieddc.xsd
```

Testing XSLT Crosswalks

The XSLT crosswalks will automatically reload an XSL stylesheet that has been modified, so you can edit and test stylesheets without restarting DSpace. You can test a dissemination crosswalk by hooking it up to an OAI-PMH crosswalk and using an OAI request to get the metadata for a known item.

Testing the submission crosswalk is more difficult, so we have supplied a command-line utility to help. It calls the crosswalk plugin to translate an XML document you submit, and displays the resulting intermediate XML (DIM). Invoke it with:

```
[dspace]/bin/dspace dsrun
     org.dspace.content.crosswalk.XSLTIngestionCrosswalk [-l] plugin input-file
```

where *plugin* is the name of the crosswalk plugin to test (e.g. "LOM"), and *input-file* is a file containing an XML document of metadata in the appropriate format.

Add the `-l` option to pass the ingestion crosswalk a list of elements instead of a whole document, as if the List form of the ingest() method had been called. This is needed to test ingesters for formats like DC that get called with lists of elements instead of a root element.

Configurable Qualified Dublin Core (QDC) dissemination crosswalk

The QDC crosswalk is a self-named plugin. To configure an instance of the QDC crosswalk, add a property to the DSpace configuration starting with "`crosswalk.qdc.properties.`"; the final word of the property name becomes the plugin's name. For example, a property name `crosswalk.qdc.properties.QDC` defines a crosswalk plugin named "`QDC`".

The following is from *dspace.cfg* file:

Properties:	`crosswalk.qdc.namspace.qdc.dc`
Example Value:	`crosswalk.qdc.namspace.qdc.dc = http://purl.org/dc/elements/1.1_`
Properties:	`crosswalk.qdc.namspace.qdc.dcterms`
Example Value:	`crosswalk.qdc.namspace.qdc.dc = http://purl.org/dc/terms/_`
Properties:	`crosswalk.qdc.schemaLocation.QDC`

Example Value:	`crosswalk.qdc.schemaLocation.QDC = http://www.purl.org/dc/terms \`
	`http://dublincore.org/schemas/xmls/qdc/2006/01/06/dcterms.xsd \`
	`http://purl.org/dc/elements/1.1 \`
	`http://dublincore.org/schemas/xmls/qdc/2006/01/06/dc.xsd`

Properties:	`crosswalk.qdc.properties.QDC`
Example Value:	`crosswalk.qdc.properties.QDC = crosswalks/QDC.properties`
Informational Note:	Configuration of the QDC Crosswalk dissemination plugin for Qualified DC. *(Add lower-case name for OAI-PMH. That is, change QDC to qdc.)*}}

In the property key "`crosswalk.qdc.properties.QDC`" the value of this property is a path to a separate properties file containing the configuration for this crosswalk. The pathname is relative to the DSpace configuration directory `/[dspace]/config` . Referring back to the "Example Value" for this property key, one has `crosswalks/qdc.properties` which defines a crosswalk named `QDC` whose configuration comes from the file `[dspace]/config/crosswalks/qdc.properties` .

You will also need to configure the namespaces and schema location strings for the XML output generated by this crosswalk. The namespaces properties names are formatted:

`crosswalk.qdc.namespace.`*prefix* = *uri*

where *prefix* is the namespace prefix and *uri* is the namespace URI. See the above Property and Example Value keys as the default dspace.cfg has been configured.

The QDC crosswalk properties file is a list of properties describing how DSpace metadata elements are to be turned into elements of the Qualified DC XML output document. The property name is a concatenation of the metadata schema, element name, and optionally the qualifier. For example, the `contributor.author` element in the native Dublin Core schema would be: `dc.contributor.author` . The value of the property is an XML fragment, the element whose value will be set to the value of the metadata field in the property key.

For example, in this property:

```
dc.coverage.temporal = <dcterms:temporal />
```

the generated XML in the output document would look like, e.g.:
`<dcterms:temporal>Fall, 2005</dcterms:temporal>`

Configuring Crosswalk Plugins

Ingestion crosswalk plugins are configured as named or self-named plugins for the interface `org.dspace.content.crosswalk.IngestionCrosswalk`. Dissemination crosswalk plugins are configured as named or self-named plugins for the interface `org.dspace.content.crosswalk.DisseminationCrosswalk`.

You can add names for existing crosswalks, add new plugin classes, and add new configurations for the configurable crosswalks as noted below.

Configuring Packager Plugins

Package ingester plugins are configured as named or self-named plugins for the interface `org.dspace.content.packager.PackageIngester` . Package disseminator plugins are configured as named or self-named plugins for the interface `org.dspace.content.packager.PackageDisseminator` .

You can add names for the existing plugins, and add new plugins, by altering these configuration properties. See the Plugin Manager architecture for more information about plugins.

Event System Configuration

If you are unfamiliar with the Event System in DSpace, and require additional information with terms like "Consumer" and "Dispatcher" please refer to:http://wiki.dspace.org/index.php/EventSystemPrototype

Property:	`event.dispatcher.default.class`
Example Value:	`event.dispatcher.default.class = org.dspace.event.BasicDispatcher`
Informational Note:	This is the default synchronous dispatcher (Same behavior as traditional DSpace).
Property:	`event.dispatcher.default.consumers`
Example Value:	`event.dispatcher.default.consumers = search, browse, eperson`
Informational Note:	This is the default synchronous dispatcher (Same behavior as traditional DSpace).
Property:	`event.dispatcher.noindex.class`
Example Value:	`event.dispatcher.noindex.class = org.dspace.event.BasicDispatcher`
Informational Note:	The noindex dispatcher will not create search or browse indexes (useful for batch item imports).
Property:	`event.dispatcher.noindex.consumers`
Example Value:	`event.dispatcher.noindex.consumers = eperson`
Informational Note:	The noindex dispatcher will not create search or browse indexes (useful for batch item imports).
Property:	`event.consumer.search.class`
Example Value:	`event.consumer.search.class = org.dspace.search.SearchConsumer`
Informational Note:	Consumer to maintain the search index.
Property:	`event.consumer.search.filters`
Example Value:	{{event.consumer.search.filters = }} `Community | Collection | Item | Bundle+Add | Create | Modify | Modify_Metadata | Delete | Remove`
Informational Note:	Consumer to maintain the search index.
Property:	`event.consumer.browse.class`
Example Value:	`event.consumer.browse.class = org.dspace.browse.BrowseConsumer`
Informational Note:	Consumer to maintain the browse index.
Property:	`event.consumer.browse.filters`
Example Value:	`event.consumer.browse.filters =` `Community | Collection | Item | Bundle+Add | Create | Modify | Modify_Metadata | Delete | Remove`

Informational Note:	Consumer to maintain the browse index.
Property:	`event.consumer.eperson.class`
Example Value:	`event.consumer.eperson.class = org.dspace.eperson.EPersonConsumer`
Informational Note:	Consumer related to EPerson changes
Property:	`event.consumer.eperson.filters`
Example Value:	`event.consumer.eperson.filters = EPerson+Create`
Informational Note:	Consumer related to EPerson changes
Property:	`event.consumer.test.class`
Example Value:	`event.consumer.test.class = org.dspace.event.TestConsumer`
Informational Note:	Test consumer for debugging and monitoring. **Commented out by default.**
Property:	`event.consumer.test.filters`
Example Value:	`event.consumer.test.filters = All+All`
Informational Note:	Test consumer for debugging and monitoring. **Commented out by default.**
Property:	`testConsumer.verbose`
Example Value:	`testConsumer.verbose = true`
Informational Note:	Set this to true to enable testConsumer messages to standard output. **Commented out by default.**

Embargo

DSpace embargoes utilize standard metadata fields to hold both the 'terms' and the 'lift date'. Which fields you use are configurable, and no specific metadata element is dedicated or predefined for use in embargo. Rather, you specify exactly what field you want the embargo system to examine when it needs to find the terms or assign the lift date.

Property:	`embargo.field.terms`
Example Value:	`embargo.field.terms = SCHEMA.ELEMENT.QUALIFIER`
Informational Note:	Embargo terms will be stored in the item metadata. This property determines in which metadata field these terms will be stored. An example could be dc.embargo.terms
Property:	`embargo.field.lift`
Example Value:	`embargo.field.lift = SCHEMA.ELEMENT.QUALIFIER`
Informational Note:	The Embargo lift date will be stored in the item metadata. This property determines in which metadata field the computed embargo lift date will be stored. You may need to create a DC metadata field in your Metadata Format Registry if it does not already exist. An example could be dc.embargo.liftdate
Property:	`embargo.terms.open`
Example Value:	`embargo.terms.open = forever`
Informational Note:	You can determine your own values for the embargo.field.terms property (see above). This property determines what the string value will be for indefinite embargos. The string in terms field to indicate indefinite embargo.
Property:	`plugin.single.org.dspace.embargo.EmbargoSetter`
Example Value:	`plugin.single.org.dspace.embargo.EmbargoSetter = org.dspace.embargo.DefaultEmbargoSetter`

Informational Note:	To implement the business logic to set your embargos, you need to override the EmbargoSetter class. If you use the value DefaultEmbargoSetter, the default implementation will be used.
Property:	`plugin.single.org.dspace.embargo.EmbargoLifter`
Example Value:	`plugin.single.org.dspace.embargo.EmbargoLifter = org.dspace.embargo.DefaultEmbargoLifter`
Informational Note:	To implement the business logic to lift your embargos, you need to override the EmbargoLifter class. If you use the value DefaultEmbargoLifter, the default implementation will be used.

More Embargo Details

More details on Embargo configuration, including specific examples can be found in the Embargo section of the documentation.

Checksum Checker Settings

DSpace now comes with a Checksum Checker script (`[dspace]/bin/dspace checker`) which can be scheduled to verify the checksum of every item within DSpace. Since DSpace calculates and records the checksum of every file submitted to it, this script is able to determine whether or not a file has been changed (either manually or by some sort of corruption or virus). The idea being that the earlier you can identify a file has changed, the more likely you'd be able to recover it (assuming it was not a wanted change).

Property:	`plugin.single.org.dspace.checker.BitstreamDispatcher`
Example Value:	`plugin.single.org.dspace.checker.BitstreamDispatcher = org.dspace.checker.SimpleDispatcher`
Informational Note:	The Default dispatcher is case non is specified.
Property:	`checker.retention.default`
Example Value:	`checker.retention.default = 10y`
Informational Note:	This option specifies the default time frame after which all checksum checks are removed from the database (defaults to 10 years). This means that after 10 years, all successful or unsuccessful matches are removed from the database.
Property:	`checker.retention.CHECKSUM_MATCH`
Example Value:	`checker.retention.CHECKSUM_MATCH = 8w`
Informational Note:	This option specifies the time frame after which a successful match will be removed from your DSpace database (defaults to 8 weeks). This means that after 8 weeks, all successful matches are automatically deleted from your database (in order to keep that database table from growing too large).

More Checksum Checking Details

For more information on using DSpace's built-in Checksum verification system, see the section on Validating CheckSums of Bitstreams.

Item Export and Download Settings

It is possible for an authorized user to request a complete export and download of a DSpace item in a compressed zip file. This zip file may contain the following:
dublin_core.xml
license.txt
contents *(listing of the contents)*
handle *file itself and the extract file if available*

The configuration settings control several aspects of this feature:

Property:	`org.dspace.app.itemexport.work.dir`
Example Value:	`org.dspace.app.itemexport.work.dir = ${dspace.dir}/exports`

Informational Note:	The directory where the exports will be done and compressed.
Property:	`org.dspace.app.itemexport.download.dir`
Example Value:	`org.dspace.app.itemexport.download.dir = ${dspace.dir}/exports/download`
Informational Note	The directory where the compressed files will reside and be read by the downloader.
Property:	`org.dspace.app.itemexport.life.span.hours`
Example Value:	`org.dspace.app.itemexport.life.span.hours = 48`
Informational Note	The length of time in hours each archive should live for. When new archives are created this entry is used to delete old ones.
Property:	`org.dspace.app.itemexport.max.size`
Example Value:	`org.dspace.app.itemexport.max.size = 200`
Informational Note	The maximum size in Megabytes (Mb) that the export should be. This is enforced before the compression. Each bitstream's size in each item being exported is added up, if their cumulative sizes are more than this entry the export is not kicked off.

Subscription Emails

DSpace, through some advanced installation and setup, is able to send out an email to collections that a user has subscribed. The user who is subscribed to a collection is emailed each time an item id added or modified. The following property key controls whether or not a user should be notified of a modification.

Property:	`eperson.subscription.onlynew`
Example Value:	`eperson.subscription.onlynew = true`
Informational Note:	For backwards compatibility, the subscription emails by default include any modified items. The property key is COMMENTED OUT by default.

Hiding Metadata

It is now possible to hide metadata from public consumption that is only available to the Administrator.

Property:	`metadata.hide.dc.description.provenance`
Example Value:	`metadata.hide.dc.description.provenance = true`
Informational Note:	Hides the metadata in the property key above except to the administrator. Fields named here are hidden in the following places UNLESS the logged-in user is an Administrator: 1. XMLUI metadata XML view, and Item splash pages (long and short views). 2. JSPUI Item splash pages 3. OAI-PMH server, "oai_dc" format. (Note: Other formats are ***not*** affected.)To designate a field as hidden, add a property here in the form: `metadata.hide.SCHEMA.ELEMENT.QUALIFIER = true`. This default configuration hides the `dc.description.provenance` field, since that usually contains email addresses which ought to be kept private and is mainly of interest to administrators.

Settings for the Submission Process

These settings control two aspects of the submission process: thesis submission permission and whether or not a bitstream file is required when submitting to a collection.

Property:	`webui.submit.blocktheses`
Example Value:	`webui.submit.blocktheses = false`

Informational Note:	Controls whether or not that the submission should be marked as a thesis.
Property:	`webui.submit.upload.required`
Example Value:	`webui.submit.upload.required = true`
Informational Note:	Whether or not a file is **required** to be uploaded during the "Upload" step in the submission process. The default is true. If set to "false", then the submitter (human being) has the option to skip the uploading of a file.

Configuring Creative Commons License

This enables the Creative Commons license step in the submission process of either the JSP or XML User Interface (JSP UI or XML UI). Submitters are given an opportunity to select a Creative Common license to accompany the item. Creative Commons licenses govern the use of the content. For further details, refer to the Creative Commons website at http://creativecommons.org . Creative Commons licensing is enabled as one step of the configurable submission process, and therefore may be configured for any given collection that has a defined submission sequence, or be part of the 'default' submission process. This process is described in the 'Customizing and Configuring Submission User Interface' section of this manual. There is a Creative Commons step already defined (step 5), but it is commented out, so enabling Creative Commons licensing is typically just a matter of uncommenting the CC License step. For the JSP UI, Creative Commons licensing is effected by opening an Iframe to the Creative Commons site and capturing the selection result in several bitstreams, but the XML UI utilizes a more flexible web service. By default, when a license is selected in the interface, the URI for the license is stored in the 'dc.rights.uri' metadata field for the Item, and a representation of the license text is stored in a license bundle. In addition, the following properties in `[dspace]/config/dspace.cfg` may be customized for use:

Property:	`cc.api.rooturl`
Example Value:	`cc.api.rooturl = http://api.creativecommons.org/rest/1.5`
Informational Note:	`Generally will never have to assign a different value - this is the base URL of the Creative Commons service API.`
Property:	`cc.license.uri`
Example Value:	`cc.license.uri = dc.rights.uri`
Informational Note:	`The field that holds the Creative Commons license URI. If you change from the default value (dc.rights.uri), you will have to reconfigure the XMLUI for proper display of license data`
Property:	`cc.license.name`
Example Value:	`cc.license.name = dc.rights`
Informational Note:	`The field that holds the Creative Commons license Name. If you change from the default value (dc.rights), you will have to reconfigure the XMLUI for proper display of license data`
Property:	`cc.submit.setname`
Example Value:	`cc.submit.setname = true`
Informational Note:	`If true, the license assignment will add the field configured with the 'cc.license.name' with the name of the CC license; if false, only 'cc.license.uri' field is added.`
Property:	`cc.submit.addbitstream`
Example Value:	`cc.submit.addbitstream = true`
Informational Note:	`If true, the license assignment will add a bitstream with the CC license RDF; if false, only metadata field(s) are added.`
Property:	`cc.license.classfilter`
Example Value:	`cc.license.classfilter = recombo,mark`
Informational Note:	`This list defines the values that will be excluded from the license (class) selection list, as defined by the web service at the URL: http://api.creativecommons.org/rest/1.5/classes`

Property:	`cc.license.jurisdiction`
Example Value:	`cc.license.jurisdiction = nz`
Informational Note:	Should a jurisdiction be used? If so, which one? See http://creativecommons.org/international/ for a list of possible codes (e.g. nz = New Zealand, uk = England and Wales, jp = Japan)

WEB User Interface Configurations

General Web User Interface Configurations
In this section of Configuration, we address the agnostic WEB User Interface that is used for JSP UI and XML UI. Some of the configurations will give information towards customization or refer you to the appropriate documentation.

Property:	webui.licence_bundle.show
Example Value:	webui.licence_bundle.show = false
Informational Note:	Sets whether to display the contents of the license bundle (often just the deposit license in the standard DSpace installation).
Property:	`webui.browse.thubnail.show`
Example Value:	`webui.browse.thubnail.show = true`
Informational Note:	Controls whether to display thumbnails on browse and search result pages. If you have customized the Browse columnlist, then you must also include a "thumbnail" column in your configuration. _(This configuration property key is not used by XMLUI. To show thumbnails using XMLUI, you need to create a theme which displays them)._
Property:	`webui.browse.thumbnail.maxheight`
Example Value:	`webui.browse.thumbnail.maxheight = 80`
Informational Note:	This property determines the maximum height of the browse/search thumbnails in pixels (px). This only needs to be set if the thumbnails are required to be smaller than the dimensions of thumbnails generated by MediaFilter.
Property:	`webui.browse.thumbnail.maxwidth`
Example Value:	`webui.browse.thumbnail.maxwidth = 80`

Informational Note:	This determines the maximum width of the browse/search thumbnails in pixels (px). This only needs to be set if the thumbnails are required to be smaller than the dimensions of thumbnails generated by MediaFilter.
Property:	`webui.item.thumbnail.show`
Example Value:	`webui.item.thumbnail.show = true`
Informational Note:	This determines whether or not to display the thumbnail against each bitstream. *(This configuration property key is not used by XMLUI. To show thumbnails using XMLUI, you need to create a theme which displays them).*
Property:	`webui.browse.thumbnail.linkbehavior`
Example Value:	`webui.browse.thumbnail.linkbehavior = item`
Informational Note:	This determines where clicks on the thumbnail in browse and search screens should lead. The only values currently supported are "item" or "bitstream", which will either take the user to the item page, or directly download the bitstream.
Property:	`thumbnail.maxwidth`
Example Value:	`thumbnail.maxwidth = 80`
Informational Note:	This property sets the maximum width of generated thumbnails that are being displayed on item pages.
Property:	`thumbnail.maxheight`
Example Value:	`thumbnail.maxheight = 80`
Informational Note:	This property sets the maximum height of generated thumbnails that are being displayed on item pages.
Property:	`webui.preview.enabled`

Example Value:	`webui.preview.enabled = false`
Informational Note:	Whether or not the user can "preview" the image.
Property:	`webui.preview.maxwidth`
Example Value:	`webui.preview.maxwidth = 600`
Informational Note:	This property sets the maximum width for the preview image.
Property:	`webui.preview.maxheight`
Example Value:	`webui.preview.maxheight = 600`
Informational Note:	This property sets the maximum height for the preview image.
Property:	`webui.preview.brand`
Example Value:	`webui.preview.brand = My Institution Name`
Informational Note:	This is the brand text that will appear with the image.
Property:	`webui.preview.brand.abbrev`
Example Value:	`webui.preview.brand.abbrev = MyOrg`

Informational Note:	An abbreviated form of the full Branded Name. This will be used when the preview image cannot fit the normal text.
Property:	`webui.preview.brand.height`
Example Value:	`webui.preview.brand.height = 20`
Informational Note:	The height (in px) of the brand.
Property:	`webui.preview.brand.font`
Example Value:	`webui.preview.brand.font = SansSerif`
Informational Note:	This property sets the font for your Brand text that appears with the image.
Property:	`webui.preview.brand.fontpoint`
Example Value:	`webui.preview.brand.fontpoint = 12`
Informational Note:	This property sets the font point (size) for your Brand text that appears with the image.
Property:	`webui.preview.dc`
Example Value:	`webui.preview.dc = rights`
Informational Note:	The Dublin Core field that will display along with the preview. This field is optional.
Property:	`webui.strengths.show`

Example Value:	`webui.strengths.show = false`
Informational Note:	Determines if communities and collections should display item counts when listed. The default behavior if omitted, is true. *(This configuration property key is not used by XMLUI. To show thumbnails using XMLUI, you need to create a theme which displays them).*
Property:	`webui.strengths.cache`
Example Value:	`webui.strengths.cache = false`
Informational Note:	When showing the strengths, should they be counted in real time, or fetched from the cache. Counts fetched in real time will perform an actual count of the database contents every time a page with this feature is requested, which will not scale. If you set the property key is set to cache ("true") you must run the following command periodically to update the count: `/[dspace]/bin/dspace itemcounter`. The default is to count in real time (set to "false").

Browse Index Configuration

The browse indexes for DSpace can be extensively configured. This section of the configuration allows you to take control of the indexes you wish to browse, and how you wish to present the results. The configuration is broken into several parts: defining the indexes, defining the fields upon which users can sort results, defining truncation for potentially long fields (e.g. authors), setting cross-links between different browse contexts (e.g. from an author's name to a complete list of their items), how many recent submissions to display, and configuration for item mapping browse.

Property:	`webui.browse.index.<n>`
Example Value:	{{webui.browse.index.1 = dateissued:metadata:dc.date.issued:date:full }}
Informational Note:	This is an example of how one "Defines the Indexes". See Defining the Indexes in the next sub-section.
Property:	`webui.itemlist.sort-option.<n>`
Example Value:	`webui.itemlist.sort-option.1 = title:dc.title:title`
Informational Note:	This is an example of how one "Defines the Sort Options". See Defining Sort Options in the following sub-section.

Defining the Indexes.

DSpace arrives with four default indexes already defined: author, title, date issued, and subjects. Users may also define additional indexes or re-configure the current indexes for different levels of specificity. For example, the default entries that appear in the *dspace.cfg* as default installation:

```
webui.browse.index.1 = dateissued:metadata:dc.date.issued:date:full
webui.browse.index.2 = author:metadata:dc.contributor.*:text
webui.browse.index.3 = title:metadata:dc.title:title:full
webui.browse.index.4 = subject:metadata:dc.subject.*:text
#webui.browse.index.5 = dateaccessioned:item:dateaccessioned
```

The format of each entry is `webui.browse.index.<n> = <index name>:<metadata>:<schema prefix>.<element>.<qualifier>:<data-type field>:<sort option>`. Please notice that the punctuation is paramount in typing this property key in the `dspace.cfg` file. The following table explains each element:

Element	Definition and Options (if available)
`webui.browse.index.<n>`	*n* is the index number. The index numbers must start from 1 and increment continuously by 1 thereafter. Deviation from this will cause an error during install or a configuration update. So anytime you add a new browse index, remember to increase the number. (Commented out index numbers may be used over again).
`<index name>`	The name by which the index will be identified. You will need to update your Messages.properties file to match this field. (The form used in the Messages.properties file is: `browse.type.metadata.<index name>` .

<metadata>	Only two options are available: "metadata" or "item"
<schema prefix>	The schema used for the field to be index. The default is dc (for Dublin Core).
<element>	The schema element. In Dublin Core, for example, the author element is referred to as "Contributor". The user should consult the default Dublin Core Metadata Registry table in Appendix A.
<qualifier>	This is the qualifier to the <element> component. The user has two choices: an asterisk "**" or a proper qualifier of the element. The asterisk is a wildcard and causes DSpace to index all types of the schema element. For example, if you have the element "contributor" and the qualifier "**" then you would index all contributor data regardless of the qualifier. Another example, you have the element "subject" and the qualifier "lcsh" would cause the indexing of only those fields that have the qualifier "lcsh". (This means you would only index Library of Congress Subject Headings and not all data elements that are subjects.
<datatype field>	This refers to the datatype of the field: date the index type will be treated as a date object title the index type will be treated like a title, which will include a link to the item page text the index type will be treated as plain text. If single mode is specified then this will link to the full mode list
<index display>	Choose full or single. This refers to the way that the index will be displayed in the browse listing. "Full" will be the full item list as specified by webui.itemlist.columns ; "single" will be a single list of only the indexed term.

If you are customizing this list beyond the default, you will need to insert the text you wish to appear in the navigation and on link and buttons. You need to edit the Messages.properties file. The form of the parameter(s) in the file:
browse.type.<index name>

Defining Sort Options

Sort options will be available when browsing a list of items (i.e. only in "full" mode, not "single" mode). You can define an arbitrary number of fields to sort on, irrespective of which fields you display using *web.itemlist.columns*. For example, the default entries that appear in the *dspace.cfg* as default installation:

```
webui.itemlist.sort-option.1 = title:dc.title:title
webui.itemlist.sort-option.2 = dateissued:dc.date.issued:date
webui.itemlist.sort-option.3 = dateaccessioned:dc.date.accessioned:date
```

The format of each entry is web.browse.sort-option.<n> = <option name>:<schema prefix>.<element>.<qualifier>:<datatype>. Please notice the punctuation used between the different elements. The following table explains the each element:

Element	Definition and Options (if available)
webui.browse.index.*n*	*n* is an arbitrary number you choose.
<option name>	The name by which the sort option will be identified. This may be used in later configuration or to locate the message key (found in *Messages.properties* file) for this index.
<schema prefix>	The schema used for the field to be index. The default is dc (for Dublin Core).
<element>	The schema element. In Dublin Core, for example, the author element is referred to as "Contributor". The user should consult the default Dublin Core Metadata Registry table in Appendix A.
<qualifier>	This is the qualifier to the <element> component. The user has two choices: an asterisk "*" or a proper qualifier of the element.
<datatype field>	This refers to the datatype of the field: date the sort type will be treated as a date object text the sort type will be treated as plain text.

Browse Index Normalization Rule Configuration

Normalization Rules are those rules that make it possible for the indexes to intermix entries without regard to case sensitivity. By default, the display of metadata in the browse indexes are case-sensitive. In the example below, you retrieve separate entries:
Twain, Marktwain, markTWAIN, MARK
However, clicking through from either of these will result in the same set of items (i.e., any item that contains either representation in the correct field).

Property:	webui.browse.metadata.case-insensitive
Example Value:	webui.browse.metadata.case-insensitive = true

Informational Note:	This controls the normalization of the index entry. Uncommenting the option (which is commented out by default) will make the metadata items case-insensitive. This will result in a single entry in the example above. However, the value displayed may be any one of the above, depending on what representation was present in the first item indexed.

At the present time, you would need to edit your metadata to clean up the index presentation.

Other Browse Options

We set other browse values in the following section.

Property:	`webui.browse.value_columns.max`
Example Value:	`webui.browse.value_columns.max = 500`
Informational Note:	This sets the options for the size (number of characters) of the fields stored in the database. The default is 0, which is unlimited size for fields holding indexed data. Some database implementations (e.g. Oracle) will enforce their own limit on this field size. Reducing the field size will decrease the potential size of your database and increase the speed of the browse, but it will also increase the chance of mis-ordering of similar fields. The values are commented out, but proposed values for reasonably performance versus result quality. This affects the size of field for the browse value (this will affect display, and value sorting)
Property:	`webui.browse.sort_columns.max`
Example Value:	`webui.browse.sort_columns.max = 200`
Informational Note:	Size of field for hidden sort columns (this will affect only sorting, not display). Commented out as default.
Property:	`webui.browse.value_columns.omission_mark`
Example Value:	`webui.browse.value_columns.omission_mark = ...`

Informational Note:	Omission mark to be placed after truncated strings in display. The default is "...".
Property:	`plugin.named.org.dspace.sort.OrderFormatDelegate`
Example Value:	`plugin.named.org.dspace.sort.OrderFormatDelegate = \` `org.dspace.sort.OrderFormatTitleMarc21=title`

Informational Note:	This sets the option for how the indexes are sorted. All sort normalizations are carried out by the OrderFormatDelegate. The plugin manager can be used to specify your own delegates for each datatype. The default datatypes (and delegates) are:
	`author = org.dspace.sort.OrderFormatAuthor`
	`title  = org.dspace.sort.OrderFormatTitle`
	`text   = org.dspace.sort.OrderFormatText`
	If you redefine a default datatype here, the configuration will be used in preferences to the default. However, if you do not explicitly redefine a datatype, then the default will still be used in addition to the datatypes you do specify. As of DSpace release 1.5.2, the multi-lingual MARC21 title ordering is configured as default, as shown in the example above. To use the previous title ordering (before release 1.5.2), comment out the configuration in your *dspace.cfg* file.

Browse Index Authority Control Configuration

Property:	`webui.browse.index.`<*n*>
Example Value:	`webui.browse.index.5 = lcAuthor:metadataAuthority:dc.contributor.author:authority`
Informational Note:	

Author (Multiple metadata value) Display

This section actually applies to any field with multiple values, but authors are the define case and example here.

Property:	`webui.browse.author-field`
Example Value:	`webui.browse.author-field = dc.contributor.*`
Informational Note:	This defines which field is the author/editor, etc. listing.

Replace `dc.contributor.*` with another field if appropriate. The field should be listed in the configuration for `webui.itemlist.columns`, otherwise you will not see its effect. It must also be defined in `webui.itemlist.columns` as being of the datatype *text* otherwise the functionality will be overridden by the specific data type feature. (This setting is not used by the XMLUI as it is controlled by your theme).

Now that we know which field is our author or other multiple metadata value field we can provide the option to truncate the number of values displayed by default. We replace the remaining list of values with "et al" or the language pack specific alternative. Note that this is just for the default, and users will have the option of changing the number displayed when they browse the results. See the following table:

Property:	`webui.browse.author-limit`
Example Value:	`webui.browse.author-limit =` <*n*>
Informational Note: | Where <*n*> is an integer number of values to be displayed. Use `-1` for unlimited (the default value).	

Links to Other Browse Contexts

We can define which fields link to other browse listings. This is useful, for example, to link an author's name to a list of just that author's items. The effect this has is to create links to browse views for the item clicked on. If it is a "single" type, it will link to a view of all the items which share that metadata element in common (i.e. all the papers by a single author). If it is a "full" type, it will link to a view of the standard full browse page, starting with the value of the link clicked on.

Property:	`webui.browse.link.`*n*
Example Value:	`webui.browse.link.1 = author:dc.contributor.*`
Informational Note:	This is used to configure which fields should link to other browse listings. This should be associated with the name of one of the browse indexes (`webui.browse.index.n`) with a metadata field listed in `webui.itemlist.columns` above. If this condition is not fulfilled, cross-linking will not work. Note also that crosslinking only works for metadata fields not tagged as `title` in `webui.itemlist.columns`.

The format of the property key is *webui.browse.link.<n> = <index name>:<display column metadata>* Please notice the punctuation used between the elements.

Element	Definition and Options (if available)
`webui.browse.link.`*n*	{{*n* is an arbitrary number you choose
`<index name>`	This need to match your entry for the index name from *webui.browse.index* property key.
`<display column metadata>`	Use the DC element (and qualifier)

Examples of some browse links used in a real DSpace installation instance:

`webui.browse.link.1 = author:dc.contributor.*`
Creates a link for all types of contributors (authors, editors, illustrators, others, etc.)
`webui.browse.link.2 = subject:dc.subject.lcsh`
Creates a link to subjects that are Library of Congress only. In this case, you have a browse index that contains only LC Subject Headings
`webui.browse.link.3 = series:dc.relation.ispartofseries`
Creates a link for the browse index "Series". Please note this is again, a customized browse index and not part of the DSpace distributed release.

Recent Submissions

This allows us to define which index to base Recent Submission display on, and how many we should show at any one time. This uses the PluginManager to automatically load the relevant plugin for the Community and Collection home pages. Values given in examples are the defaults supplied in *dspace.cfg*

Property:	`recent.submission.sort-option`
Example Value:	`recent.submission.sort-option = dateaccessioned`
Informational Note:	First is to define the sort name (from *webui.browse.sort-options*) to use for displaying recent submissions.
Property:	`recent.submissions.count`
Example Value:	`recent.submissions.count = 5`
Informational Note:	Defines how many recent submissions should be displayed at any one time.

There will be the need to set up the processors that the PluginManager will load to actually perform the recent submissions query on the relevant pages. This is already configured by default *dspace.cfg* so there should be no need for the administrator/programmer to worry about this.

```
plugin.sequence.org.dspace.plugin.CommunityHomeProcessor = \
        org.dspace.app.webui.components.RecentCommunitySubmissions

plugin.sequence.org.dspace.plugin.CollectionHomeProcessor = \
        org.dspace.app.webui.components.RecentCollectionSubmissions
```

Submission License Substitution Variables

Property:	`plugin.named.org.dspace.content.license.`
	`LicenseArgumentFormatter`
	
	(property key broken up for display purposes only)

Example Value:	`plugin.named.org.dspace.content.license.LicenseArgumentFormatter = \`
	`org.dspace.content.license.SimpleDSpaceObjectLicenseFormatter = collection, \`
	`org.dspace.content.license.SimpleDSpaceObjectLicenseFormatter = item, \`
	`org.dspace.content.license.SimpleDSpaceObjectLicenseFormatter = eperson`

Informational Note:	It is possible include contextual information in the submission license using substitution variables. The text substitution is driven by a plugin implementation.

Syndication Feed (RSS) Settings

This will enable syndication feeds, links display on community and collection home pages. This setting is not used by the XMLUI, as you enable feeds in your theme.

Property:	`webui.feed.enable`
Example Value:	`webui.feed.enable = true`
Informational Note:	By default, RSS feeds are set to true (on) . Change key to "false" to disable.
Property:	`webui.feed.items`
Example Value:	`webui.feed.items = 4`
Informational Note:	Defines the number of DSpace items per feed (the most recent submissions)
Property:	`webui.feed.cache.size`
Example Value:	`webui.feed.cache.size = 100`
Informational Note:	Defines the maximum number of feeds in memory cache. Value of "`0`" will disable caching.
Property:	`webui.feed.cache.age`

Example Value:	`webui.feed.cache.age = 48`
Informational Note:	Defines the number of hours to keep cached feeds before checking currency. The value of "*0*" will force a check with each request.
Property:	`webui.feed.formats`
Example Value:	`webui.feed.formats = rss_1.0,rss_2.0,atom_1.0`
Informational Note:	Defines which syndication formats to offer. You can use more than one; use a comma-separated list. The following list are the available values: rss_0.90, rss_0.91, rss_0.92, rss_0.93, rss_0.94, rss_1.0, rss_2.0, atom_1.0.
Property:	`webui.feed.localresolve`
Example Value:	`webui.feed.localresolve = false`
Informational Note:	By default, (set to false), URLs returned by the feed will point at the global handle resolver (e.g. http://hdl.handle.net/123456789/1). If set to ***true*** the local server URLs are used (e.g. http://myserver.myorg/handle/123456789/1).
Property:	`webui.feed.item.title`
Example Value:	`webui.feed.item.title = dc.title`
Informational Note:	This property customizes each single-value field displayed in the feed information for each item. Each of the fields takes a **single** metadata field. The form of the key is <scheme prefix>.<element>.<qualifier> In place of the qualifier, one may leave it blank to exclude any qualifiers or use the wildcard "*" to include all qualifiers for a particular element.
Property:	`webui.feed.item.date`
Example Value:	`webui.feed.item.date = dc.date.issued`

Informational Note:	This property customizes each single-value field displayed in the feed information for each item. Each of the fields takes a **single** metadata field. The form of the key is <scheme prefix>.<element>.<qualifier> In place of the qualifier, one may leave it blank to exclude any qualifiers or use the wildcard "*" to include all qualifiers for a particular element.
Property:	`webui.feed.item.description`

Example Value:	`webui.feed.item.description = dc.title, dc.contributor.author, \`
	`dc.contributor.editor, dc.description.abstract, \`
	`dc.description`

Informational Note:	One can customize the metadata fields to show in the feed for each item's description. Elements are displayed in the order they are specified in *dspace.cfg*.Like other property keys, the format of this property key is: *webui.feed.item.description = <scheme prefix>.<element>.<qualifier>*. In place of the qualifier, one may leave it blank to exclude any qualifiers or use the wildcard "*" to include all qualifiers for a particular element.
Property:	`webui.feed.item.author`
Example Value:	`webui.feed.item.author = dc.contributor.author`
Informational Note:	The name of field to use for authors (Atom only); repeatable.
Property:	`webui.feed.logo.url`
Example Value:	`webui.feed.logo.url = ${dspace.url}/themes/mysite/images/mysite-logo.png`
Informational Note:	Customize the image icon included with the site-wide feeds. This must be an absolute URL.
Property:	`webui.feed.item.dc.creator`
Example Value:	`webui.feed.item.dc.creator = dc.contributor.author`
Informational Note:	This optional property adds *structured* DC elements as XML elements to the feed description. They are not the same thing as, for example, *webui.feed.item.description*. Useful when a program or stylesheet will be transforming a feed and wants separate author, description, date, etc.
Property:	`webui.feed.item.dc.date`

Example Value:	`webui.feed.item.dc.date = dc.date.issued`
Informational Note:	This optional property adds *structured* DC elements as XML elements to the feed description. They are not the same thing as, for example, *webui.feed.item.description*. Useful when a program or stylesheet will be transforming a feed and wants separate author, description, date, etc.
Property:	`webui.feed.item.dc.description`
Example Value:	`webui.feed.item.dc.description = dc.description.abstract`
Informational Note:	This optional property adds *structured* DC elements as XML elements to the feed description. They are not the same thing as, for example, *webui.feed.item.description*. Useful when a program or stylesheet will be transforming a feed and wants separate author, description, date, etc.
Property:	`webui.feed.podcast.collections`
Example Value:	`webui.feed.podcast.collections = 1811/45183,1811/47223`
Informational Note:	This optional property enables Podcast Support on the RSS feed for the specified collection handles. The podcast is iTunes compatible and will expose the bitstreams in the items for viewing and download by the podcast reader. Multiple values are separated by commas. For more on using /enabling Media RSS Feeds to share content via iTunesU, see: Enable Media RSS Feeds
Property:	`webui.feed.podcast.communities`
Example Value:	`webui.feed.podcast.communities = 1811/47223`
Informational Note:	This optional property enables Podcast Support on the RSS feed for the specified community handles. The podcast is iTunes compatible and will expose the bitstreams in the items for viewing and download by the podcast reader. Multiple values are separated by commas. For more on using /enabling Media RSS Feeds to share content via iTunesU, see: Enable Media RSS Feeds
Property:	`webui.feed.podcast.mimetypes`
Example Value:	`webui.feed.podcast.mimetypes = audio/x-mpeg,application/pdf`

Informational Note:	This optional property for Podcast Support, allows you to choose which MIME types of bitstreams are to be enclosed in the podcast feed. Multiple values are separated by commas. For more on using/enabling Media RSS Feeds to share content via iTunesU, see: Enable Media RSS Feeds
Property:	`webui.feed.podcast.sourceuri`
Example Value:	`webui.feed.podcast.sourceuri = dc.source.uri`
Informational Note:	This optional property for the Podcast Support will allow you to use a value for a metadata field as a replacement for actual bitstreams to be enclosed in the RSS feed. A use case for specifying the external sourceuri would be if you have a non-DSpace media streaming server that has a copy of your media file that you would prefer to have the media streamed from. For more on using/enabling Media RSS Feeds to share content via iTunesU, see: Enable Media RSS Feeds

OpenSearch Support

OpenSearch is a small set of conventions and documents for describing and using "search engines", meaning any service that returns a set of results for a query. See extensive description in the *Business Layer section* of the documentation.

Please note that for result data formatting, OpenSearch uses Syndication Feed Settings (RSS). So, even if Syndication Feeds **are not** enable, they **must** be configured to enable OpenSearch. OpenSearch uses all the configuration properties for DSpace RSS to determine the mapping of metadata fields to feed fields. Note that a new field for authors has been added (used in Atom format only).

Property:	`websvc.opensearch.enable`
Example Value:	`websvc.opensearch.enable = false`
Informational Note:	Whether or not OpenSearch is enabled. By default, the feature is disabled. Change the property key to 'true' to enable.
Property:	`websvc.opensearch.uicontext`
Example Value:	`websvc.opensearch.uicontext = simple-search`
Informational Note:	`Context for HTML request URLs. Change only for non-standard servlet mapping.`
Property:	`websvc.opensearch.svccontext`
Example Value:	`websvc.opensearch.svccontext = open-search/`
Informational Note:	Context for RSS/Atom request URLs. Change only for non-standard servlet mapping.
Property:	`websvc.opensearch.autolink`
Example Value:	`websvc.opensearch.autolink = true`
Informational Note:	Present autodiscovery link in every page head.
Property:	`websvc.opensearch.validity`
Example Value:	`websvc.opensearch.validity = 48`
Informational Note:	Number of hours to retain results before recalculating. This applies to the Manakin interface only.
Property:	`websvc.opensearch.shortname`
Example Value:	`websvc.opensearch.shortname = DSpace`

Informational Note:	A short name used in browsers for search service. It should be sixteen (16) or fewer characters.
Property:	`websvc.opensearch.longname`
Example Value:	`websvc.opensearch.longname = ${dspace.name}`
Informational Note:	A longer name up to 48 characters.
Property:	`websvc.opensearch.description`
Example Value:	`websvc.opensearch.description = ${dspace.name} DSpace repository`
Informational Note:	`Brief service description`
Property:	`websvc.opensearch.faviconurl`
Example Value:	_websvc.opensearch.faviconurl = http://www.dspace.org/images/favicon.ico_
Informational Note:	Location of favicon for service, if any. They must by 16 x 16 pixels. You can provide your own local favicon instead of the default.
Property:	`websvc.opensearch.samplequery`
Example Value:	`websvc.opensearch.samplequery = photosynthesis`
Informational Note:	Sample query. This should return results. You can replace the sample query with search terms that should actually yield results in your repository.
Property:	`websvc.opensearch.tags`
Example Value:	`websc.opensearch.tags = IR DSpace`
Informational Note:	Tags used to describe search service.
Property:	`websvc.opensearch.formats`
Example Value:	`websvc.opensearch.formats = html,atom,rss`
Informational Note:	Result formats offered. Use one or more comma-separated from the list: html, atom, rss. Please note that html is required for auto discovery in browsers to function, and must be the first in the list if present.

Content Inline Disposition Threshold

The following configuration is used to change the disposition behavior of the browser. That is, when the browser will attempt to open the file or download it to the user-specified location. For example, the default size is 8MB. When an item being viewed is larger than 8MB, the browser will download the file to the desktop (or wherever you have it set to download) and the user will have to open it manually.

Property:	`webui.content_disposition_threshold`
Example value:	`webui.content_disposition_threshold = 8388608`
Informational Note:	The default value is set to 8MB. This property key applies to the JSPUI interface.
Property:	`xmlui.content_disposition_threshold`
Example Value:	`xmlui.content_disposition_threshold = 8388608`
Informational Note:	The default value is set to 8MB. This property key applies to the XMLUI (Manakin) interface.

Other values are possible:
4 MB = 41943048 MB = 838860816 MB = 16777216

Multi-file HTML Document/Site Settings

The setting is used to configure the "depth" of request for html documents bearing the same name.

Property:	`webui.html.max-depth-guess`
Example Value:	`webui.html.max-depth-guess = 3`
Informational Note:	When serving up composite HTML items in the JSP UI, how deep can the request be for us to serve up a file with the same name? For example, if one receives a request for "*foo/bar/index.html*" and one has a bitstream called just "*index.html*", DSpace will serve up the former bitstream (*foo/bar/index.html*) for the request if *webui.html.max-depth-guess* is 2 or greater. If *webui.html.max-depth-guess* is 1 or less, then DSpace would not serve that bitstream, as the depth of the file is greater. If webui.html.max-depth-guess is zero, the request filename and path must always exactly match the bitstream name. The default is set to 3.
Property:	`xmlui.html.max-depth-guess`
Example Value:	`xmlui.html.max-depth-guess = 3`
Informational Note:	When serving up composite HTML items in the XMLUI, how deep can the request be for us to serve up a file with the same name? For example, if one receives a request for "*foo/bar/index.html*" and one has a bitstream called just "*index.html*", DSpace will serve up the former bitstream (*foo/bar/index.html*) for the request if *webui.html.max-depth-guess* is 2 or greater. If *xmlui.html.max-depth-guess* is 1 or less, then DSpace would not serve that bitstream, as the depth of the file is greater. If _webui.html.max-depth-guess _is zero, the request filename and path must always exactly match the bitstream name. The default is set to 3.

Sitemap Settings

To aid web crawlers index the content within your repository, you can make use of sitemaps.

Property:	`sitemap.dir`
Example Value:	`sitemap.dir = ${dspace.dir}/sitemaps`
Informational Note:	The directory where the generate sitemaps are stored.
Property:	`sitemap.engineurls`

Example Value:	_sitemap.engineurls = http://www.google.com/webmasters/sitemaps/ping?sitemap=_
Informational Note:	Comma-separated list of search engine URLs to 'ping' when a new Sitemap has been created. Include everything except the Sitemap UL itself (which will be URL-encoded and appended to form the actual URL 'pinged').Add the following to the above parameter if you have an application ID with Yahoo: *http://search.yahooapis.com/SiteExplorererService/V1/updateNotification?appid=REPLACE_ME?url=_. (Replace the component _REPLACE_ME* with your application ID). There is no known 'ping' URL for MSN/Live search.

Authority Control Settings

Two features fall under the header of Authority Control: Choice Management and Authority Control of Item ("DC") metadata values. Authority control is a fully optional feature in DSpace 1.6. Implemented out of the box are the Library of Congress Names service, and the Sherpa Romeo authority plugin.

For an in-depth description of this feature, please consult: Authority Control of Metadata Values

Property:	`plugin.named.org.dspace.content.authority.ChoiceAuthority`

Example Value:	`plugin.named.org.dspace.content.authority.ChoiceAuthority = \`
	`org.dspace.content.authority.SampleAuthority = Sample, \`
	`org.dspace.content.authority.LCNameAuthority = LCNameAuthority, \`
	`org.dspace.content.authority.SHERPARoMEOPublisher = SRPublisher, \`
	`org.dspace.content.authority.SHERPARoMEOJournalTitle = SRJournalTitle`

Informational Note:	`--`
Property:	`plugin.selfnamed.org.dspace.content.authority.ChoiceAuthority`
Example Value:	`plugin.selfnamed.org.dspace.content.authority.ChoiceAuthority = \` `org.dspace.content.authority.DCInputAuthority`
Property:	`lcname.url`
Example Value:	`lcname.url = http://alcme.oclc.org/srw/search/lcnaf_`
Informational Note:	Location (URL) of the Library of Congress Name Service
Property:	`sherpa.romeo.url`
Example Value:	`sherpa.romeo.url = http://www.sherpa.ac.uk/romeo/api24.php_`
Informational Note:	Location (URL) of the SHERPA/RoMEO authority plugin
Property:	`authority.minconfidence`
Example Value:	`authority.minconfidence = ambiguous`

Informational Note:	This sets the default lowest confidence level at which a metadata value is included in an authority-controlled browse (and search) index. It is a symbolic keyword, one of the following values (listed in descending order): accepted, uncertain, ambiguous, notfound, failed, rejected, novalue, unset. See `org.dspace.content.authority.Choices` source for descriptions.
Property:	`xmlui.lookup.select.size`
Example Value:	`xmlui.lookup.select.size = 12`
Informational Note:	This property sets the number of selectable choices in the Choices lookup popup

JSPUI Upload File Settings

To alter these properties for the XMLUI, please consult the Cocoon specific configuration at /WEB-INF/cocoon/properties/core.properties.

Property:	`upload.temp.dir`
Example Value:	`upload.temp.dir = ${dspace.dir}/upload`
Informational Note:	This property sets where DSpace temporarily stores uploaded files.
Property:	`upload.max`
Example Value:	`upload.max = 536870912`
Informational Note:	Maximum size of uploaded files in bytes. A negative setting will result in no limit being set. The default is set for 512Mb.

JSP Web Interface (JSPUI) Settings

The following section is limited to JSPUI. If the user wishes to use XMLUI settings, please refer to Chapter 7: XMLUI Configuration and Customization.

Property:	`webui.itemdisplay.default`

Example Value:	`webui.itemdisplay.default = dc.title, dc.title.alternative, \`
	`dc.contributor.*, dc.subject, dc.data.issued(date), \`
	`dc.publisher, dc.identifier.citation, \`
	`dc.relation.ispartofseries, dc.description.abstract, \`
	`dc.description, dc.identifier.govdoc, \`
	`dc.identifier.uri(link), dc.identifier.isbn, \`
	`dc.identifier.issn, dc.identifier.ismn, dc.identifier`

Informational Note:	This is used to customize the DC metadata fields that display in the item display (the brief display) when pulling up a record. The format is: `<schema>.<element>.<_optional_qualifier>` . In place of the qualifier, one can use the wildcard "*" to include all fields of the same element, or, leave it blank for unqualified elements. Additionally, two additional options are available for behavior/rendering: (date) and (link). See the following examples:
	
	`dc.title` = Dublin Core element 'title' (unqualified)
	`dc.title.alternative` = DC element 'title', qualifier 'alternative'
	`dc.title.*` = All fields with Dublin Core element 'title' (any or no qualifier)
	`dc.identifier.uri(link)` = DC identifier.uri, rendered as a link
	`dc.date.issued(date)` = DC date.issued, rendered as a date
	The `Messages.properties` file controls how the fields defined above will display to the user. If the field is missing from the _Messages. properties_ file, it will not be displayed. Look in `Messages.properties}}under {{metadata.dc.<field>`. Example:
	`metadata.dc.contributor.other = Authors`
	`metadata.dc.contributor.author = Authors`
	`metadata.dc.title.* = Title`
	Please note: The order in which you place the values to the property key control the order in which they will display to the user on the outside world. (See the Example Value above).

Property:	`webui.resolver.1.urn`
	`webui.resolver.1.baseurl`
	`webui.resolver.2.urn`
	`webui.resolver.2.baseurl`

Example Value:	`webui.resolver.1.urn = doi`
	`webui.resolver.1.baseurl = http://dx.doi.org/`
	`webui.resolver.2.urn = hdl`
	`webui.resolver.2.baseurl = http://hdl.handle.net/`

Informational Note:	When using "resolver" in *webui.itemdisplay* to render identifiers as resolvable links, the base URL is take from <code>webui.resolver.<n>.baseurl<code> where <code>webui.resolver.<n>.baseurl<code> matches the urn specified in the metadata value. The value is appended to the "baseurl" as is, so the baseurl needs to end with the forward slash almost in any case. If no urn is specified in the value it will be displayed as simple text. For the doi and hdl urn defaults values are provided, respectively http://dc.doi.org and http://hdl.handle.net are used. If a metadata value with style "doi", "handle" or "resolver" matches a URL already, it is simply rendered as a link with no other manipulation.
Property:	`plugin.single.org.dspace.app.webui.util.StyleSelection`

Example Value:	`plugin.single.org.dspace.app.webui.util.StyleSelection = \`
	`  org.dspace.app.web.util.CollectionStyleSelection`
	`  #org.dspace.app.web.util.MetadataStyleSelection`

Informational Note:	Specify which strategy to use for select the style for an item.
Property:	`webui.itemdisplay.thesis.collections`

Example Value:	`webui.itemdisplay.thesis.collections = 123456789/24, 123456789/35`
Informational Note:	Specify which collections use which views by Handle.
Property:	`webui.itemdisplay.metadata-style` `webui.itemdisplay.metadata-style`
Example Value:	`webui.itemdisplay.metadata-style = schema.element[.qualifier|.*]` `webui.itemdisplay.metadata-style = dc.type`
Informational Note:	Specify which metadata to use as name of the style
Property:	`webui.itemlist.columns`
Example Value:	`webui.itemlist.columns = thumbnail, dc.date.issued(date), dc.title, \` `dc.contributor.*`

Informational Note:	Customize the DC fields to use in the item listing page. Elements will be displayed left to right in the order they are specified here. The form is <schema prefix>.<element>[.<qualifier> | .*][(date)], ... Although not a requirement, it would make sense to include among the listed fields at least the date and title fields as specified by the `webui.browse.index` configuration options in the next section mentioned. (cf.) If you have enabled thumbnails (webui.browse.thumbnail.show), you must also include a 'thumbnail' entry in your columns, this is where the thumbnail will be displayed.
Property:	`webui.itemlist.width`
Example Value:	`webui.itemlist.width = *, 130, 60%, 40%`
Informational Note:	You can customize the width of each column with the following line--you can have numbers (pixels) or percentages. For the 'thumbnail' column, a setting of '*' will use the max width specified for browse thumbnails (cf. `webui.browse.thumbnail.maxwidth, thumbnail.maxwidth`)

Property:	`webui.itemlist.browse.<index name>.sort.<sort name>.columns`
	`webui.itemlist.sort.<sort name>.columns`
	`webui.itemlist.browse.<browse name>.columns`
	`webui.itemlist.<sort or index name>.columns`

Example Value:	_}}
Informational Note:	You can override the DC fields used on the listing page for a given browse index and/or sort option. As a sort option or index may be defined on a field that isn't normally included in the list, this allows you to display the fields that have been indexed/sorted on. There are a number of forms the configuration can take, and the order in which they are listed below is the priority in which they will be used (so a combination of an index name and sort name will take precedence over just the browse name).In the last case, a sort option name will always take precedence over a browse index name. Note also, that for any additional columns you list, you will need to ensure there is an *itemlist.<field name>* entry in the messages file.

Property:	`webui.itemlist.dateaccessioned.columns`
ExampleValue:	`webui.itemlist.dateaccessioned.columns = thumbnail, dc.date.accessioned(date), dc.title, dc.contributor.*`
InformationalNote:	This would display the date of the accession in place of the issue date whenever the dateaccessioned browsed index or sort option is selected. Just like *webui.itemlist.columns*, you will need to include a 'thumbnail' entry to display the thumbnails in the item list.
Property:	`webui.itemlist.dateaccessioned.widths`
ExampleValue:	`webui.itemlist.dateaccessioned.widths = *, 130, 60%, 40%`
InformationalNote:	As in the aforementioned property key, you can customize the width of the columns for each configured column list, substituting '.*widths*' for '.*columns*' in the property name. See the setting for *webui.itemlist.widths* for more information.
Property:	`webui.itemlist.tablewidth`

Example Value:	`webui.itemlist.tablewidth = 100%`
Informational Note:	You can also set the overall size of the item list table with the following setting. It can lead to faster table rendering when used with the column widths above, but not generally recommended.
Property:	`webui.session.invalidate`
Example Value:	`webui.session.invalidate = true`
Informational Note:	Enable or disable session invalidation upon login or logout. This feature is enabled by default to help prevent session hijacking but may cause problems for shibboleth, etc. If omitted, the default value is '*true*'. [Only used for JSPUI authentication].

JSPUI Configuring Multilingual Support

[i18n – Locales]

Setting the Default Language for the Application

Property:	`default.locale`
Example Value:	`default.locale = en`
Informational Note:	The default language for the application is set with this property key. This is a locale according to i18n and might consist of country, country_language or country_language_variant. If no default locale is defined, then the server default locale will be used. The format of a local specifier is described here: http://java.sun.com/j2se/1.4.2/docs/api/java/util/Locale.html

Supporting More Than One Language

Changes in dspace.cfg

Property:	`webui.supported.locale`
Example Value:	`webui.supported.locale = en, de`
`or perhaps`	`webui.supported.locals = en, en_ca, de`
Informational Note:	All the locales that are supported by this instance of DSpace. Comma separated list.

The table above, if needed and is used will result in:

- a language switch in the default header
- the user will be enabled to choose his/her preferred language, this will be part of his/her profile
- wording of emails
    - mails to registered users, e.g. alerting service will use the preferred language of the user
    - mails to unregistered users, e.g. suggest an item will use the language of the session
- according to the language selected for the session, using dspace-admin Edit News will edit the news file of the language according to session

Related Files

If you set webui.supported.locales make sure that all the related additional files for each language are available. *LOCALE* should correspond to the locale set in *webui.supported.locales*, e. g.: for webui.supported.locales = en, de, fr, there should be:

- `[dspace-source]/dspace/modules/jspui/src/main/resources/Messages.properties`
- `[dspace-source]/dspace/modules/jspui/src/main/resources/Messages_en.properties`
- `[dspace-source]/dspace/modules/jspui/src/main/resources/Messages_de.properties`
- `[dspace-source]/dspace/modules/jspui/src/main/resources/Messages_fr.properties`
  Files to be localized:

- `[dspace-source]/dspace/modules/jspui/src/main/resources/Messages_LOCALE.properties`
- `[dspace-source]/dspace/config/input-forms_LOCALE.xml`
- `[dspace-source]/dspace/config/default_LOCALE.license - should be pure ASCII`
- `[dspace-source]/dspace/config/news-top_LOCALE.html`
- `[dspace-source]/dspace/config/news-side_LOCALE.html`
- `[dspace-source]/dspace/config/emails/change_password_LOCALE`
- `[dspace-source]/dspace/config/emails/feedback_LOCALE`
- `[dspace-source]/dspace/config/emails/internal_error_LOCALE`
- `[dspace-source]/dspace/config/emails/register_LOCALE`
- `[dspace-source]/dspace/config/emails/submit_archive_LOCALE`
- `[dspace-source]/dspace/config/emails/submit_reject_LOCALE`
- `[dspace-source]/dspace/config/emails/submit_task_LOCALE`
- `[dspace-source]/dspace/config/emails/subscription_LOCALE`
- `[dspace-source]/dspace/config/emails/suggest_LOCALE`
- `[dspace]/webapps/jspui/help/collection-admin_LOCALE.html - in html keep the jump link as original; must be copied to [dspace-source]/dspace/modules/jspui/src/main/webapp/help`
- `[dspace]/webapps/jspui/help/index_LOCALE.html - must be copied to [dspace-source]/dspace/modules/jspui/src/main/webapp/help`
- `[dspace]/webapps/jspui/help/site-admin_LOCALE.html - must be copied to [dspace-source]/dspace/modules/jspui/src/main/webapp/help`

JSPUI Item Mapper

Because the item mapper requires a primitive implementation of the browse system to be present, we simply need to tell that system which of our indexes defines the author browse (or equivalent) so that the mapper can list authors' items for mapping

Define the index name (from *webui.browse.index*) to use for displaying items by author.

Property:	`itemmap.author.index`
Example Value:	`itemmap.author.index = author`
Informational Note:	If you change the name of your author browse field, you will also need to update this property key.

Display of Group Membership

Property:	`webui.mydspace.showgroupmembership`
Example Value:	`webui.mydspace.showgroupmembership = false`
Informational Note:	To display group membership set to "true". If omitted, the default behavior is false.

JSPUI / XMLUI SFX Server

SFX Server is an OpenURL Resolver.

Property:	`sfx.server.url`
Example Value:	`sfx.server.url` = `http://sfx.myu.edu:8888/sfx?`
	`sfx.server.url` = `http://worldcatlibraries.org/registry/gateway?`
Informational Note:	SFX query is appended to this URL. If this property is commented out or omitted, SFX support is switched off.

All the parameters mapping are defined in `[dspace]/config/sfx.xml` file. The program will check the parameters in `sfx.xml` and retrieve the correct metadata of the item. It will then parse the string to your resolver.

For the following example, the program will search the first query-pair which is DOI of the item. If there is a DOI for that item, your retrieval results will be, for example:
`http://researchspace.auckland.ac.nz/handle/2292/5763`

Example. For setting DOI in `sfx.xml`

```
<query-pairs>
          <field>
                <querystring>rft_id=info:doi/</querystring>
                <dc-schema>dc</dc-schema>
                <dc-element>identifier</dc-element>
                <dc-qualifier>doi</dc-qualifier>
          </field>
        </query-pairs>
```

If there is no DOI for that item, it will search next query-pair based on the `[dspace]/config/sfx.xml` and then so on.

Example of using ISSN, volume, issue for item without DOI
`[http://researchspace.auckland.ac.nz/handle/2292/4947]`

For parameter passing to the `<querystring>`

```
<querystring>rft_id=info:doi/</querystring>
```

Please refer to these:
`[http://ocoins.info/cobgbook.html]`
`[http://ocoins.info/cobg.html]`

Program assume won't get empty string for the item, as there will at least author, title for the item to pass to the resolver.

For contributor author, program maintains original DSpace SFX function of extracting author's first and last name.

```
<field>
            <querystring>rft.aulast=</querystring>
            <dc-schema>dc</dc-schema>
            <dc-element>contributor</dc-element>
                    <dc-qualifier>author</dc-qualifier>
         </field>
         <field>
                  <querystring>rft.aufirst=</querystring>
            <dc-schema>dc</dc-schema>
          <dc-element>contributor</dc-element>
          <dc-qualifier>author</dc-qualifier>
         </field>
```

JSPUI Item Recommendation Setting

Property:	`webui.suggest.enable`
Example Value:	webui.suggest.enable = true
Informational Note:	Show a link to the item recommendation page from item display page.
Property:	`webui.suggest.loggedinusers.only`
Example Value:	`webui.suggest.loggedinusers.only = true`
Informational Note:	Enable only if the user is logged in. If this key commented out, the default value is false.

Controlled Vocabulary Settings

DSpace now supports controlled vocabularies to confine the set of keywords that users can use while describing items.

Property:	`webui.controlledvocabulary.enable`
Example Value:	`webui.controlledvocabulary.enable = true`
Informational Note:	Enable or disable the controlled vocabulary add-on. WARNING: This feature is not compatible with WAI (it requires JavaScript to function).

The need for a limited set of keywords is important since it eliminates the ambiguity of a free description system, consequently simplifying the task of finding specific items of information.

The controlled vocabulary add-on allows the user to choose from a defined set of keywords organized in an tree (taxonomy) and then use these keywords to describe items while they are being submitted.

We have also developed a small search engine that displays the classification tree (or taxonomy) allowing the user to select the branches that best describe the information that he/she seeks.

The taxonomies are described in XML following this (very simple) structure:

```
<node id="acmccs98" label="ACMCCS98">
    <isComposedBy>
        <node id="A." label="General Literature">
            <isComposedBy>
                <node id="A.0" label="GENERAL"/>
                <node id="A.1" label="INTRODUCTORY AND SURVEY"/>
            </isComposedBy>
        </node>
    </isComposedBy>
</node>
```

You are free to use any application you want to create your controlled vocabularies. A simple text editor should be enough for small projects. Bigger projects will require more complex tools. You may use Protegé to create your taxonomies, save them as OWL and then use a XML Stylesheet (XSLT) to transform your documents to the appropriate format. Future enhancements to this add-on should make it compatible with standard schemas such as OWL or RDF.

In order to make DSpace compatible with WAI 2.0, the add-on is **turned off** by default (the add-on relies strongly on JavaScript to function). It can be activated by setting the following property in `dspace.cfg`:

```
webui.controlledvocabulary.enable = true
```

New vocabularies should be placed in `[dspace]/config/controlled-vocabularies/` and must be according to the structure described. A validation XML Schema (named `controlledvocabulary.xsd`) is also available in that directory.

Vocabularies need to be associated with the correspondent DC metadata fields. Edit the file `[dspace]/config/input-forms.xml` and place a *"vocabulary"* tag under the *"field"* element that you want to control. Set value of the *"vocabulary"* element to the name of the file that contains the vocabulary, leaving out the extension (the add-on will only load files with extension "*.xml"). For example:

```
<field>
    <dc-schema>dc</dc-schema>
    <dc-element>subject</dc-element>
    <dc-qualifier></dc-qualifier>
    <!-- An input-type of twobox MUST be marked as repeatable -->
    <repeatable>true</repeatable>
    <label>Subject Keywords</label>
    <input-type>twobox</input-type>
    <hint> Enter appropriate subject keywords or phrases below. </hint>
    <required></required>
    <vocabulary [closed="false"]>nsi</vocabulary>
</field>
```

The vocabulary element has an optional boolean attribute **closed** that can be used to force input only with the javascript of controlled-vocabulary add-on. The default behavior (i.e. without this attribute) is as set **closed="false"**. This allow the user also to enter the value in free way.

The following vocabularies are currently available by default:

- **nsi** - *nsi.xml* - The Norwegian Science Index
- **srsc** - *srsc.xml* - Swedish Research Subject Categories

3. JSPUI Session Invalidation

Property:	`webui.session.invalidate`
Example Value:	`webui.session.invalidate = true`
Informational Note:	Enable or disable session invalidation upon login or logout. This feature is enabled by default to help prevent session hijacking but may cause problems for shibboleth, etc. If omitted, the default value is 'true'.

XMLUI Specific Configuration

The DSpace digital repository supports two user interfaces: one based upon JSP technologies and the other based upon the Apache Cocoon framework. This section describes those configurations settings which are specific to the XMLUI interface based upon the Cocoon framework. *(Prior to DSpace Release 1.5.1 XMLUI was referred to Manakin. You may still see references to "Manakin")*

Property:	`xmlui.supported.locales`
Example Value:	`xmlui.supported.locales = en, de`
Informational Note:	A list of supported locales for Manakin. Manakin will look at a user's browser configuration for the first language that appears in this list to make available to in the interface. This parameter is a comma separated list of Locales. All types of Locales country, country_language, country_language_variant. Note that if the appropriate files are not present (i.e. Messages_XX_XX.xml) then Manakin will fall back through to a more general language.
Property:	`xmlui.force.ssl`
Example Value:	`xmlui.force.ssl = true`
Informational Note:	Force all authenticated connections to use SSL, only non-authenticated connections are allowed over plain http. If set to true, then you need to ensure that the '*dspace.hostname*' parameter is set to the correctly.
Property:	`xmlui.user.registration`
Example Value:	`xmlui.user.registration = true`
Informational Note:	Determine if new users should be allowed to register. This parameter is useful in conjunction with Shibboleth where you want to disallow registration because Shibboleth will automatically register the user. Default value is true.
Property:	`xmlui.user.editmetadata`

Example Value:	`xmlui.user.editmetadata = true`
Informational Note:	Determines if users should be able to edit their own metadata. This parameter is useful in conjunction with Shibboleth where you want to disable the user's ability to edit their metadata because it came from Shibboleth. Default value is true.
Property:	`xmlui.user.assumelogon`
Example Value:	`xmlui.user.assumelogon = true`
Informational Note:	Determine if super administrators (those whom are in the Administrators group) can login as another user from the "edit eperson" page. This is useful for debugging problems in a running dspace instance, especially in the workflow process. The default value is false, i.e., no one may assume the login of another user.
Property:	`xmlui.user.loginredirect`
Example Value:	`xmlui.user.loginredirect = /profile`
Informational Note:	After a user has logged into the system, which url should they be directed? Leave this parameter blank or undefined to direct users to the homepage, or */profile* for the user's profile, or another reasonable choice is */submissions* to see if the user has any tasks awaiting their attention. The default is the repository home page.
Property:	`xmlui.theme.allowoverrides`
Example Value:	`xmlui.theme.allowoverrides = false`
Informational Note:	Allow the user to override which theme is used to display a particular page. When submitting a request add the HTTP parameter "themepath" which corresponds to a particular theme, that specified theme will be used instead of the any other configured theme. Note that this is a potential security hole allowing execution of unintended code on the server, this option is only for development and debugging it should be turned off for any production repository. The default value unless otherwise specified is "false".
Property:	`xmlui.bundle.upload`
Example Value:	`xmlui.bundle.upload = ORIGINAL, METADATA, THUMBNAIL, LICENSE, CC_LICENSE`

Informational Note:	Determine which bundles administrators and collection administrators may upload into an existing item through the administrative interface. If the user does not have the appropriate privileges (add and write) on the bundle then that bundle will not be shown to the user as an option.
Property:	`xmlui.community-list.render.full`
Example Value:	`xmlui.community-list.render.full = true`
Informational Note:	On the community-list page should all the metadata about a community/collection be available to the theme. This parameter defaults to true, but if you are experiencing performance problems on the community-list page you should experiment with turning this option off.
Property:	`xmlui.community-list.cache`
Example Value:	`xmlui.community-list.cache = 12 hours`
Informational Note:	Normally, Manakin will fully verify any cache pages before using a cache copy. This means that when the community-list page is viewed the database is queried for each community/collection to see if their metadata has been modified. This can be expensive for repositories with a large community tree. To help solve this problem you can set the cache to be assumed valued for a specific set of time. The downside of this is that new or editing communities/collections may not show up the website for a period of time.
Property:	`xmlui.bistream.mods`
Example Value:	`xmlui.bistream.mods = true`
Informational Note:	Optionally, you may configure Manakin to take advantage of metadata stored as a bitstream. The MODS metadata file must be inside the "METADATA" bundle and named MODS.xml. If this option is set to 'true' and the bitstream is present then it is made available to the theme for display.
Property:	`xmlui.bitstream.mets`
Example Value:	`xmlui.bitstream.mets = true`
Informational Note:	Optionally, you may configure Manakin to take advantage of metadata stored as a bitstream. The METS metadata file must be inside the "METADATA" bundle and named METS.xml. If this option is set to "true" and the bitstream is present then it is made available to the theme for display.

Property:	`xmlui.google.analytics.key`
Example Value:	`xmlui.google.analytics.key = UA-XXXXXX-X`
Informational Note:	If you would like to use Google Analytics to track general website statistics then use the following parameter to provide your analytics key. First sign up for an account at http://analytics.google.com, then create an entry for your repositories website. Google Analytics will give you a snippet of javascript code to place on your site, inside that snip it is your Google Analytics key usually found in the line: _uacct = "UA-XXXXXXX-X" Take this key (just the UA-XXXXXX-X part) and place it here in this parameter.
Property:	`xmlui.controlpanel.activity.max`
Example Value:	`xmlui.controlpanel.activity.max = 250`
Informational Note:	Assign how many page views will be recorded and displayed in the control panel's activity viewer. The activity tab allows an administrator to debug problems in a running DSpace by understanding who and how their dspace is currently being used. The default value is 250.
Property:	`xmlui.controlpanel.activity.ipheader`
Example Value:	`xmlui.controlpanel.activity.ipheader = X-Forward-For`
Informational Note:	Determine where the control panel's activity viewer receives an events IP address from. If your DSpace is in a load balanced environment or otherwise behind a context-switch then you will need to set the parameter to the HTTP parameter that records the original IP address.

DSpace SOLR Statistics Configuration

Property:	`solr.log.server`
Example Value:	`solr.log.server = ${dspace.baseUrl}/solr/statistics`
Informational Note:	Is used by the SolrLogger Client class to connect to the SOLR server over http and perform updates and queries.
Property:	`solr.spidersfile`
Example Value:	`solr.spidersfile = ${dspace.dir}/config/spiders.txt`

Informational Note:	Spiders file is utilized by the SolrLogger, this will be populated by running the following command:`dsrun org.dspace.statistics.util.SpiderDetector -i <httpd log file>`
Property:	`solr.dbfile`
Example Value:	`solr.dbfile = ${dspace.dir}/config/GeoLiteCity.dat`
Informational Note:	The following refers to the GeoLiteCity database file utilized by the *LocationUtils* to calculate the location of client requests based on IP address. During the Ant build process (both *fresh_install* and *update*) this file will be downloaded from http://www.maxmind.com/app/geolitecity if a new version has been published or it is absent from your `[dspace]/config` directory.
Property:	`solr.resolver.timeout`
Example Value:	`solr.resolver.timeout = 200`
Informational Note:	Timeout for the resolver in the DNS lookup time in milliseconds, defaults to 200 for backward compatibility; your system's default is usually set in `/etc/resolv.conf` and varies between 2 to 5 seconds, too high a value might result in solr exhausting your connection pool.
Property:	`statistics.item.authorization.admin`
Example Value:	`statistics.item.authorization.admin = true`
Informational Note:	Enables access control restriction on DSpace Statistics pages, Restrictions are based on access rights to Community, Collection and Item Pages. This will require the user to sign on to see that statistics. Setting the statistics to "false" will make them publicly available.
Property:	`solr.statistics.logBots`
Example Value:	{{solr.statistics.logBots = true }}
Informational Note:	Enable/disable logging of spiders in solr statistics. If false, and IP matches an address in solr.spiderips.urls, event is not logged. If true, event will be logged with the 'isBot' field set to true (see `solr.statistics.query.filter.*` for query filter options) Default value is true.
Property:	`solr.statistics.query.filter.spiderIp`
Example Value:	`solr.statistics.query.filter.spiderIp = false`
Informational Note:	Controls solr statistics querying to filter out spider IPs. False by default.
Property:	{{solr.statistics.query.filter.isBot }}
Example Value:	`solr.statistics.query.filter.isBot = true`
Informational Note:	Controls solr statistics querying to look at "isBot" field to determine if record is a bot. True by default.
Property:	`solr.spiderips.urls`

Example Value:	`solr.spiderips.urls = http://iplists.com/google.txt, \`
	`http://iplists.com/inktomi.txt, \`
	`http://iplists.com/lycos.txt, \`
	`http://iplists.com/infoseek.txt, \`
	`http://iplists.com/altavista.txt,\`
	`http://iplists.com/excite.txt, \`
	`http://iplists.com/misc.txt, \`
	`http://iplists.com/excite.txt, \`
	`http://iplists.com/misc.txt, \`
	`http://iplists.com/non_engines.txt`

Informational Note:	URLs to download IP addresses of search engine spiders from

Optional or Advanced Configuration Settings

The following section explains how to configure either optional features or advanced features that are not necessary to make DSpace "out-of-the-box"

The Metadata Format and Bitstream Format Registries

The *[dspace]/config/registries* directory contains three XML files. These are used to load the *initial* contents of the Dublin Core Metadata registry and Bitstream Format registry and SWORD metadata registry. After the initial loading (performed by *ant fresh_install* above), the registries reside in the database; the XML files are not updated.

In order to change the registries, you may adjust the XML files before the first installation of DSpace. On an already running instance it is recommended to change bitstream registries via DSpace admin UI, but the metadata registries can be loaded again at any time from the XML files without difficult. The changes made via admin UI are not reflected in the XML files.

Metadata Format Registries

The default metadata schema is Dublin Core, so DSpace is distributed with a default Dublin Core Metadata Registry. Currently, the system requires that every item have a Dublin Core record.

There is a set of Dublin Core Elements, which is used by the system and should not be removed or moved to another schema, see Appendix: Default Dublin Core Metadata registry.

Note: altering a Metadata Registry has no effect on corresponding parts, e.g. item submission interface, item display, item import and vice versa. Every metadata element used in submission interface or item import must be registered before using it.

Note also that deleting a metadata element will delete all its corresponding values.

If you wish to add more metadata elements, you can do this in one of two ways. Via the DSpace admin UI you may define new metadata elements in the different available schemas. But you may also modify the XML file (or provide an additional one), and re-import the data as follows:

```
[dspace]/bin/dspace dsrun org.dspace.administer.MetadataImporter -f [xml file]
```

The XML file should be structured as follows:

```
<dspace-dc-types>
    <dc-type>
        <schema>dc</schema>
        <element>contributor</element>
        <qualifier>advisor</qualifier>
        <scope_note>Use primarily for thesis advisor.</scope_note>
    </dc-type>
</dspace-dc-types>
```

Bitstream Format Registry

The bitstream formats recognized by the system and levels of support are similarly stored in the bitstream format registry. This can also be edited at install-time via *[dspace]/config/registries/bitstream-formats.xml* or by the administration Web UI. The contents of the bitstream format registry are entirely up to you, though the system requires that the following two formats are present:

- *Unknown*

- *License*
  Deleting a format will cause any existing bitstreams of this format to be reverted to the unknown bitstream format.

XPDF Filter

This is an alternative suite of MediaFilter plugins that offers faster and more reliable text extraction from PDF Bitstreams, as well as thumbnail image generation. It replaces the built-in default PDF MediaFilter.

If this filter is so much better, why isn't it the default? The answer is that it relies on external executable programs which must be obtained and installed for your server platform. This would add too much complexity to the installation process, so it left out as an optional "extra" step.

Installation Overview

Here are the steps required to install and configure the filters:

1. Install the xpdf tools for your platform, from the downloads at http://www.foolabs.com/xpdf
2. Acquire the Sun Java Advanced Imaging Tools and create a local Maven package.
3. Edit DSpace configuration properties to add location of xpdf executables, reconfigure MediaFilter plugins.
4. Build and install DSpace, adding -Pxpdf-mediafilter-support to Maven invocation.

Install XPDF Tools

First, download the XPDF suite found at: http://www.foolabs.com/xpdf and install it on your server. The executables can be located anywhere, but make a note of the full path to each command.

You may be able to download a binary distribution for your platform, which simplifies installation. Xpdf is readily available for Linux, Solaris, MacOSX, Windows, NetBSD, HP-UX, AIX, and OpenVMS, and is reported to work on AIX, OS/2, and many other systems.

The only tools you *really* need are:

- *pdfinfo* - displays properties and Info dict
- *pdftotext* - extracts text from PDF
- *pdftoppm* - images PDF for thumbnails

Fetch and install jai_imageio JAR

Fetch and install the Java Advanced Imaging Image I/O Tools.

For AIX, Sun support has the following: "JAI has native acceleration for the above but it also works in pure Java mode. So as long as you have an appropriate JDK for AIX (1.3 or later, I believe), you should be able to use it. You can download any of them, extract just the jars, and put those in your $CLASSPATH."

Download the *jai_imageio* library version 1.0_01 or 1.1 found at: https://jai-imageio.dev.java.net/binary-builds.html#Stable_builds .

For these filters you do NOT have to worry about the native code, just the JAR, so choose a download for any platform.

```
curl -O http://download.java.net/media/jai-imageio/builds/release/1.1/jai_imageio-1_1-lib-linux-i586.tar.gz
tar xzf jai_imageio-1_1-lib-linux-i586.tar.gz
```

The preceding example leaves the JAR in ***jai_imageio-1_1/lib/jai_imageio.jar*** . Now install it in your local Maven repository, e.g.: (changing the path after *file*= if necessary)

```
mvn install:install-file                       \
          -Dfile=jai_imageio-1_1/lib/jai_imageio.jar  \
          -DgroupId=com.sun.media                     \
          -DartifactId=jai_imageio                    \
          -Dversion=1.0_01                            \
          -Dpackaging=jar                             \
          -DgeneratePom=true
```

You may have to repeat this procedure for the *jai_core.jar* library, as well, if it is not available in any of the public Maven repositories. Once acquired, this command installs it locally:

```
mvn install:install-file -Dfile=jai_core-1.1.2_01.jar  \
    -DgroupId=javax.media -DartifactId=jai_core -Dversion=1.1.2_01 -Dpackaging=jar -DgeneratePom=true
```

Edit DSpace Configuration

First, be sure there is a value for *thumbnail.maxwidth* and that it corresponds to the size you want for preview images for the UI, e.g.: (*NOTE:* this code doesn't pay any attention to *thumbnail.maxheight* but it's best to set it too so the other thumbnail filters make square images.)

```
# maximum width and height of generated thumbnails
        thumbnail.maxwidth= 80
        thumbnail.maxheight = 80
```

Now, add the absolute paths to the XPDF tools you installed. In this example they are installed under */usr/local/bin* (a logical place on Linux and MacOSX), but they may be anywhere.

```
xpdf.path.pdftotext = /usr/local/bin/pdftotext
        xpdf.path.pdftoppm = /usr/local/bin/pdftoppm
        xpdf.path.pdfinfo = /usr/local/bin/pdfinfo
```

Change the MediaFilter plugin configuration to remove the old *org.dspace.app.mediafilter.PDFFilter* and add the new filters, e.g: (New sections are in bold)

```
filter.plugins = \
        PDF Text Extractor, \
        PDF Thumbnail, \
        HTML Text Extractor, \
        Word Text Extractor, \
        JPEG Thumbnail
         plugin.named.org.dspace.app.mediafilter.FormatFilter = \
        org.dspace.app.mediafilter.XPDF2Text = PDF Text Extractor, \
        org.dspace.app.mediafilter.XPDF2Thumbnail = PDF Thumbnail, \
        org.dspace.app.mediafilter.HTMLFilter = HTML Text Extractor, \
        org.dspace.app.mediafilter.WordFilter = Word Text Extractor, \
        org.dspace.app.mediafilter.JPEGFilter = JPEG Thumbnail, \
        org.dspace.app.mediafilter.BrandedPreviewJPEGFilter = Branded Preview JPEG
```

Then add the input format configuration properties for each of the new filters, e.g.:

```
filter.org.dspace.app.mediafilter.XPDF2Thumbnail.inputFormats = Adobe PDF
filter.org.dspace.app.mediafilter.XPDF2Text.inputFormats = Adobe PDF
```

Finally, if you want PDF thumbnail images, don't forget to add that filter name to the *filter.plugins* property, e.g.:

```
filter.plugins = PDF Thumbnail,  PDF Text Extractor, ...
```

Build and Install

Follow your usual DSpace installation/update procedure, only add *-Pxpdf-mediafilter-support* to the Maven invocation:

```
mvn -Pxpdf-mediafilter-support package
     ant -Dconfig=\[dspace\]/config/dspace.cfg update
```

Configuring Usage Instrumentation Plugins

A usage instrumentation plugin is configured as a singleton plugin for the abstract class org.dspace.app.statistics.AbstractUsageEvent.

The Passive Plugin

The Passive plugin is provided as the class org.dspace.app.statistics.PassiveUsageEvent. It absorbs events without effect. Use the Passive plugin when you have no use for usage event postings. This is the default if no plugin is configured.

The Tab File Logger Plugin

The Tab File Logger plugin is provided as the class org.dspace.app.statistics.UsageEventTabFileLogger. It writes event records to a file in tab-separated column format. If left unconfigured, an error will be noted in the DSpace log and no file will be produced. To specify the file path, provide an absolute path as the value for usageEvent.tabFileLogger.file in dspace.cfg.

The XML Logger Plugin

The XML Logger plugin is provided as the class org.dspace.app.statistics.UsageEventXMLLogger. It writes event records to a file in a simple XML-like format. If left unconfigured, an error will be noted in the DSpace log and no file will be produced. To specify the file path, provide an absolute path as the value for usageEvent.xmlLogger.file in dspace.cfg.