XMLUI Configuration and Customization

DSpace System Documentation: Manakin [XMLUI] Configuration and Customization

The DSpace digital repository supports two user interfaces: one based on JavaServer Pages (JSP) technologies and one based upon the Apache Cocoon framework (XMLUI). This chapter describes those parameters which are specific to the Manakin (XMLUI) interface based upon the Cocoon framework.

1 Manakin Configuration Property Keys
2 Configuring Themes and Aspects
    2.1 Aspects
    2.2 Themes
3 Multilingual Support
4 Creating a New Theme
5 Customizing the News Document
6 Adding Static Content
7 Enabling OAI-ORE Harvester using XMLUI
    7.1 Automatic Harvesting (Scheduler)
8 Additional XMLUI Learning Resources

Manakin Configuration Property Keys

In an effort to save the programmer/administrator some time, the configuration table below is taken from 5.3.43. *XMLUI Specific Configuration*.

Property:	*xmlui.supportedLocales*
Example Value:	*xmlui.supportedLocales = 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 snipit 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.

Configuring Themes and Aspects

The Manakin user interface is composed of two distinct components: *aspects* and *themes*. Manakin aspects are like extensions or plugins for Manakin; they are interactive components that modify existing features or provide new features for the digital repository. Manakin themes stylize the look-and-feel of the repository, community, or collection.

The repository administrator is able to define which aspects and themes are installed for the particular repository by editing the *[dspace]/config/xmlui.xconf* configuration file. The *xmlui.xconf* file consists of two major sections: Aspects and Themes.

Aspects

The <*aspects*> section defines the "Aspect Chain", or the linear set of aspects that are installed in the repository. For each aspect that is installed in the repository, the aspect makes available new features to the interface. For example, if the "submission" aspect were to be commented out or removed from the *xmlui.xconf*, then users would not be able to submit new items into the repository (even the links and language prompting users to submit items are removed). Each <*aspect*> element has two attributes, *name* and *path*. The name is used to identify the Aspect, while the path determines the directory where the aspect's code is located. Here is the default aspect configuration:

```
<aspects>
    <aspect name="Artifact Browser" path="resource://aspects/ArtifactBrowser/" />
    <aspect name="Administration"        path="resource://aspects/Administrative/" />
    <aspect name="E-Person" path="resource://aspects/EPerson/"        />
    <aspect name="Submission and Workflow" path="resource://aspects/Submission/" />
</aspects>
```

A standard distribution of Manakin/DSpace includes four "core" aspects:

- **Artifact Browser** The Artifact Browser Aspect is responsible for browsing communities, collections, items and bitstreams, viewing an individual item and searching the repository.
- **E-Person** The E-Person Aspect is responsible for logging in, logging out, registering new users, dealing with forgotten passwords, editing profiles and changing passwords.
- **Submission** The Submission Aspect is responsible for submitting new items to DSpace, determining the workflow process and ingesting the new items into the DSpace repository.
- **Administrative** The Administrative Aspect is responsible for administrating DSpace, such as creating, modifying and removing all communities, collections, e-persons, groups, registries and authorizations.

Themes

The <*themes*> section defines a set of "rules" that determine where themes are installed in the repository. Each rule is processed in the order that it appears, and the first rule that matches determines the theme that is applied (so order is important). Each rule consists of a <*theme*> element with several possible attributes:

- **name** (*always required*)The name attribute is used to document the theme's name.
- **path** (*always required*)The path attribute determines where the theme is located relative to the *themes/* directory and must either contain a trailing slash or point directly to the theme's *sitemap.xmap* file.
- **regex** (*either regex and/or handle is required*)The regex attribute determines which URLs the theme should apply to.

- **handle** (*either regex and/or handle is required*)The handle attribute determines which community, collection, or item the theme should apply to. If you use the "handle" attribute, the effect is cascading, meaning if a rule is established for a community then all collections and items within that community will also have this theme apply to them as well. Here is an example configuration:

```
<themes>
    <theme name="Theme 1" handle="123456789/23" path="theme1/"/>
    <theme name="Theme 2" regex="community-list"        path="theme2/"/>
    <theme name="Reference Theme" regex=".*" path="Reference/"/>
</themes>
```

  In the example above three themes are configured: "Theme 1", "Theme 2", and the "Reference Theme". The first rule specifies that "Theme 1" will apply to all communities, collections, or items that are contained under the parent community "123456789/23". The next rule specifies any URL containing the string "community-list" will get "Theme 2". The final rule, using the regular expression ".**", will match *anything**, so all pages which have not matched one of the preceding rules will be matched to the Reference Theme.

Multilingual Support

The XMLUI user interface supports multiple languages through the use of internationalization catalogues as defined by the Cocoon Internationalization Transformer. Each catalog contains the translation of all user-displayed strings into a particular language or variant. Each catalog is a single xml file whose name is based upon the language it is designated for, thus:

messages_*language_country_variant*.xml

messages_*language_country*.xml

messages_*language*.xml

messages.xml

The interface will automatically determine which file to select based upon the user's browser and system configuration. For example, if the user's browser is set to Australian English then first the system will check if *messages_en_au.xml* is available. If this translation is not available it will fall back to *messages_en.xml*, and finally if that is not available, *messages.xml*.

Manakin supplies an English only translation of the interface. In order to add other translations to the system, locate the *[dspace-source]/dspace/modules/xmlui/src/main/webapp/i18n/* directory. By default this directory will be empty; to add additional translations add alternative versions of the *messages.xml* file in specific language and country variants as needed for your installation.

To set a language other than English as the default language for the repository's interface, simply name the translation catalogue for the new default language "*messages.xml*"

Creating a New Theme

Manakin themes stylize the look-and-feel of the repository, community, or collection and are distributed as self-contained packages. A Manakin/DSpace installation may have multiple themes installed and available to be used in different parts of the repository. The central component of a theme is the sitemap.xmap, which defines what resources are available to the theme such as XSL stylesheets, CSS stylesheets, images, or multimedia files.

1) Create theme skeleton

Most theme developers do not create a new theme from scratch; instead they start from the standard theme template, which defines a skeleton structure for a theme. The template is located at: *[dspace-source]/dspace-xmlui/dspace-xmlui-webbapp/src/main/webbapp/themes/template*. To start your new theme simply copy the theme template into your locally defined modules directory, *[dspace-source]/dspace/modules/xmlui/src/main/webbapp/themes/[your theme's directory]/*.

2) Modify theme variables

The next step is to modify the theme's parameters so that the theme knows where it is located. Open the *[your theme's directory]/sitemap.xmap* and look for <*global-variables*>

```
<global-variables>
    <theme-path>[your theme's         directory]</theme-path>
    <theme-name>[your theme's name]</theme-name>
</global-variables>
```

Update both the theme's path to the directory name you created in step one. The theme's name is used only for documentation.

3) Add your CSS stylesheets

The base theme template will produce a repository interface without any style - just plain XHTML with no color or formatting. To make your theme useful you will need to supply a CSS Stylesheet that creates your desired look-and-feel. Add your new CSS stylesheets:

[your theme's directory]/lib/style.css (The base style sheet used for all browsers)

[your theme's directory]/lib/style-ie.css (Specific stylesheet used for internet explorer)

4) Install theme and rebuild DSpace

Next rebuild and deploy DSpace (replace <version> with the your current release):

1. Rebuild the DSpace installation package by running the following command from your *[dspace-source]/dspace/* directory:

```
mvn package
```

2. Update all DSpace webapps to *[dspace]/webapps* by running the following command from your *[dspace-source]/dspace/target/dspace-[version]-build.dir* directory:

```
ant -Dconfig=[dspace]/config/dspace.cfg update
```

3. Deploy the the new webapps:

```
cp -R /[dspace]/webapps/* /[tomcat]/webapps
```

4. Restart Tomcat
   This will ensure the theme has been installed as described in the previous section "Configuring Themes and Aspects".

Customizing the News Document

The XMLUI "news" document is only shown on the root page of your repository. It was intended to provide the title and introductory message, but you may use it for anything.

The news document is located at *[dspace]/dspace/config/news-xmlui.xml*. There is only one version; it is localized by inserting "i18n" callouts into the text areas. It must be a complete and valid XML DRI document (see Chapter 15).

Its (the News document) exact rendering in the XHTML UI depends, of course, on the theme. The default content is designed to operate with the reference themes, so when you modify it, be sure to preserve the tag structure and e.g. the exact attributes of the first DIV tag. Also note that the text is DRI, not HTML, so you must use only DRI tags, such as the XREF tag to construct a link.

Example 1: a single language:

```
<document>
      <body>
        <div id="file.news.div.news" n="news" rend="primary">
          <head> TITLE OF YOUR REPOSITORY HERE </head>
          <p>
            INTRO MESSAGE HERE
            Welcome to my wonderful repository etc etc ...
            A service of <xref target="http://myuni.edu/">My University</xref>
          </p>
        </div>
      </body>
      <options/>
      <meta>
        <userMeta/>
        <pageMeta/>
        <repositoryMeta/>
      </meta>
   </document>
```

Example 2: all text replaced by references to localizable message keys:

```
<document>
      <body>
        <div id="file.news.div.news" n="news" rend="primary">
          <head><i18n:text>myuni.repo.title</i18n:text></head>
          <p>
            <i18n:text>myuni.repo.intro</i18n:text>
            <i18n:text>myuni.repo.a.service.of</i18n:text>
            <xref target="http://myuni.edu/"><i18n:text>myuni.name</i18n:text></xref>
          </p>
        </div>
      </body>
      <options/>
      <meta>
        <userMeta/>
        <pageMeta/>
        <repositoryMeta/>
      </meta>
    </document>
```

Adding Static Content

The XMLUI user interface supports the addition of globally static content (as well as static content within individual themes).

Globally static content can be placed in the *[dspace-source]/dspace/modules/xmlui/src/main/webapp/static/* directory. By default this directory only contains the default *robots.txt* file, which provides helpful site information to web spiders/crawlers. However, you may also add static HTML (**.html*) content to this directory, as needed for your installation.

Any static HTML content you add to this directory may also reference static content (e.g. CSS, Javascript, Images, etc.) from the same *[dspace-source] /dspace/modules/xmlui/src/main/webapp/static/* directory. You may reference other static content from your static HTML files similar to the following:

```
<link href="./static/mystyle.css" rel="stylesheet" type="text/css"/>
<img src="./static/images/static-image.gif" alt="Static image in /static/images/ directory"/>
<img src="./static/static-image.jpg" alt="Static image in /static/ directory"/>
```

Enabling OAI-ORE Harvester using XMLUI

This section will give the necessary steps to set up the OAI-ORE Harvester usig Manakin.

Setting up a collection (Collection Edit Screen):

1. Login and create a new collection.
2. Go to the tab named "Content Source" that now appears next to "Edit Metadata" and "Assign Roles " in the collection edit screens.
3. The two counter source options are standards (selected by default) and harvested. Select "harvests from external source" and click Save.
4. A new set of menus appear to configure the harvesting settings:
    - "OAI Provide" is in the URL of the OAI-PMH provider that the content from this collection should be harvested from. The PMH provider deployed with DSpace typically has the form:"http://dspace.url/oai/reuqest". For this example use "http://web01.library.tamu.edu/oai-h151 /request"
    - "OAI Set id" is the setSpec of the collection you wish to harvest from.Use "hdl_1969.1_5671" for this example.
    - "Metadata format" determines the format that the descriptive metdata will be harvested. Since DSpace stores metadata in its own internal format, not all metadata values might bet harvested if a specific format is specified. Select "DSpace Intermediate Metadata" if available and "Simple Dublin Core" otherwise.
    - Click the Test Settings button will verify the settings supplied in the previous steps and will usually let you know what, if anything is missing or does not match up.
5. The list of radio buttons labeled "Content being harvested" allows you to select the harvest level. The first one requires no OAI-ORE support on the part of the provider and can be used to harvest metadata from any provider compliant with the OAI-PMH 2.0 specifications. The middle options will harvest the metadata and generate links to bitstreams stored remotely, while the last one will perform full local replication. Select the middle option and click Save
   At this point the settings are saved and the menu changes to provide three options:

- "Change Settings" takes you back to the edit screen.
- "Import Now" performs a single harvest from the remote collection into the local one. Success, notes, and errors encountered in the process will be reflected in the "Last Harvest Result" entry. More detailed information is available in the DSpace log. Note that the whole harvest cycle is executed within a single HTTP request and will time out for large collections. For this reason, it is advisable to use the automatic harvest scheduler set up
  either in XMLUI or from the command line. If the scheduler is running, "Import Now" will handle the harvest task as a separate thread.
- "Reset and Reimport Collection" will perform the same function as "Import Now", but will clear the collection of all existing items before doing so.

Automatic Harvesting (Scheduler)

Setting up automatic harvesting in the Control Panel Screen.

- A new table, Harvesting, has been added under Administrative > Control Panel.
- The panel offers the following information:
  - Available actions:
    - Start Harvester: starts the scheduler. From this point on, all properly configured collections (listed on the next line) will be harvested at regular intervals. This interval can be changed in the *dspace.cfg* using the "*harvester.harvestFrequency*" parameter.
    - Pause: the "nice" stop; waits for the active harvests to finish, saves the state/progress and pauses execution. Can be either resumed or stopped.
    - Stop: the "full stop"; waits for the current item to finish harvesting, and aborts further execution.
    - Reset Harvest Status: since stopping in the middle of a harvest is likely to result in collections getting "stuck" in the queue, the button is available to clear all states.

Additional XMLUI Learning Resources

Useful links with further information into XMLUI Development

- Making DSpace XMLUI Your Own - Concentrates on using Maven to build Overlays in the XMLUI (Manakin). Also has very basic examples for JSPUI. Based on DSpace 1.6.x.
- Learning to Use Manakin (XMLUI) - Overview of how to use Manakin and how it works. Based on DSpace 1.5, but also valid for 1.6.
- Introducing Manakin (XMLUI)