

Bungeni Editor Operations Manual

last updated : 19th August 2009

Table of Contents

About the Editor.....	3
Installing the Editor.....	4
Installation Steps for Ubuntu.....	4
Pre-requisites.....	4
Sun Java 6 JRE.....	4
OpenOffice.org 3.0.1.....	4
Step 1 – Open the installer using the Sun Java 6 runtime	5
Step 2 – Provides a description of what will be installed and what is required for a successful installation.....	6
Step 3 – Information about the Project.....	6
Step 4 – Licensing terms and conditions.....	7
Step 5 – Select an installation directory.....	7
Step 6 – Agree to installation.....	8
Step 7 - Informational screen – click next to install	8
Step 8 – Installation progress and completion.....	9
Step 9 – Installation completion screen.....	9
Installation Steps for Windows.....	10
Pre-requisites.....	10
Sun Java 6 JRE.....	10
OpenOffice.org 3.0.1.....	10
Editor Installation.....	10
Caveats.....	10
Classloader errors	10
Editor Architecture.....	11
The Editor User Interface.....	12
The Editing Process – Control Panel.....	13
Step 1 Compose New / Edit Existing.....	13
Step 2 Metadata Panel.....	14
Step 3 – Validate and Check document	15
Step 4 – Transform to different formats.....	16
The Editing Process – Markup Panel.....	17
Starting the Editor.....	18
Customizing and Extending the Editor.....	19
Editor Folder Structure.....	19
Document Types.....	19
Document Templates.....	19
Settings Database.....	20
UI Theme.....	20
i18n – Internationalization & Localization.....	21
Editor Markup Actions.....	21
Editor Plugins.....	22
Introduction.....	22
Plugin Definition.....	22
Required Interfaces.....	23
Building the Editor from Source.....	23
Pre-requisites.....	23
Building from source.....	23

About the Editor

The Bungeni Editor is a java application that extends the OpenOffice.org 3.0 word processor. The editor provides a hybrid UI with markup tools around the main word processor window.

One of the main objectives of the editor is to simplify the legislative document creation process. As legislative users are familiar with a word processor interface – the bungeni editor extends this UI paradigm by building functionality around the word processor window.

“Markup” tools are provided by the extended UI to allow the user to identify parts of a legislative document and associate data fragments (“metadata”) with different parts of the document. The ultimate objective of going through the effort of marking up a document is to produce legislative XML. The bungeni editor provides a facility to export marked up word processor documents in both AkomaNtoso and MetaLEX formats.

The Bungeni Editor also provides a assistive UI to help the user create proper markup. Markup tools and icons are contextually highlighted and enabled to indicate valid markup actions to the user. Apart from the assistive UI the Bungeni Editor also provides two levels of markup validation – a semantic validation layer which provides checks at the OpenOffice.org word processor document level and an Xml validation layer which checks the transformed document against an Xml schema.

The Bungeni Editor can in theory work on any operating system that supports OpenOffice.org v 3.0, however it has been primarily tested on the Ubuntu Linux and Windows XP platforms.

Installing the Editor

Detailed installation steps are provided on the Editor installation wiki page :

<http://code.google.com/p/bungeni-editor/wiki/InstallingTheBungeniEditor>

Graphical instructions are provided below.

Installation Steps for Ubuntu

Pre-requisites

The editor requires the following software as pre-requisites :

Sun Java 6 JRE

Ubuntu by default comes packaged with the GCJ open source JRE. The Editor will not work on this JRE as it requires the Swing UI libraries that come packaged with the Sun JRE release. Install the Sun JRE on the Ubuntu OS either through package manager. The Editor requires at least JRE / JDK version 1.6 + . To install the Sun Java JRE on Ubuntu open a terminal window run the following:

```
sudo apt-get install sun-java6-jre
```

OpenOffice.org 3.0.1

Ubuntu comes packaged with an OpenOffice release. Unfortunately this OpenOffice installation is a repackaged installation produced by Ubuntu and it changes the folder structure of the OpenOffice installation. This change makes it incompatible with the cross platform UNO class loader used by the Bungeni Editor.

As a preliminary step, uninstall the packaged OpenOffice provided by ubuntu by uninstalling it using package manager – you may also do so by running the following command from a terminal window :

```
sudo apt-get remove openoffice*
```

Download the OpenOffice 3.0.1 debian package from :

http://openoffice.mirrors.tds.net/pub/openoffice/stable/3.0.1/OOo_3.0.1_LinuxIntel_install_en-US_deb.tar.gz

Download the installation package and extract it a folder on your computer.

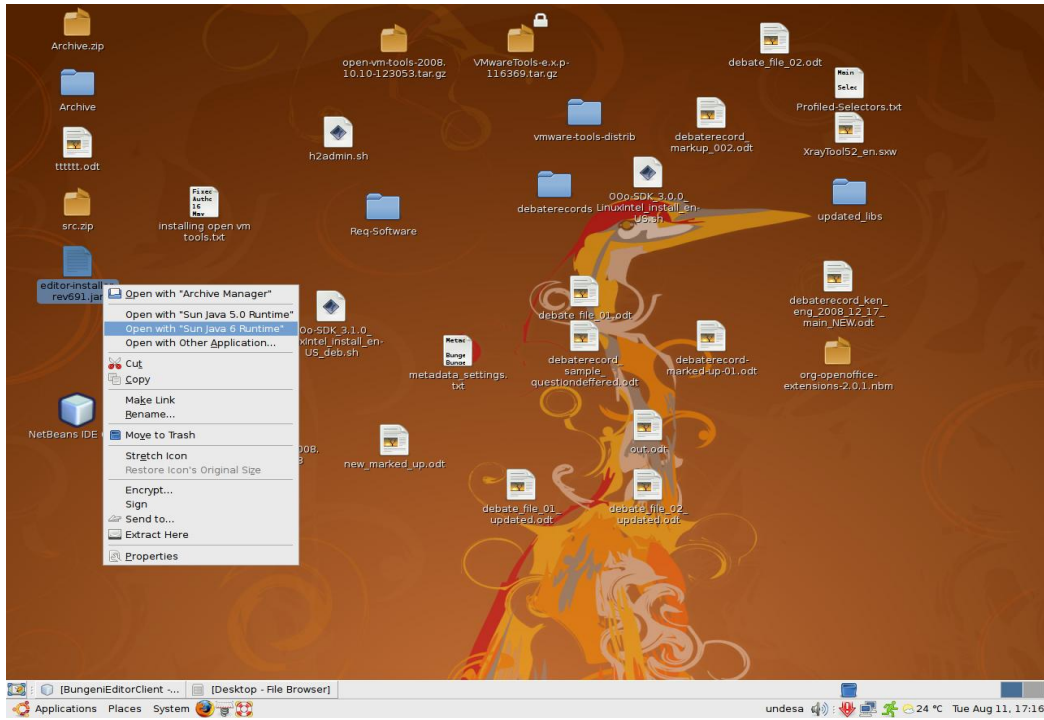
Install the package by running the following command from the openoffice-extracted-folder/DEBS path :

```
sudo dpkg -i *.deb
```

Install the desktop integration package :

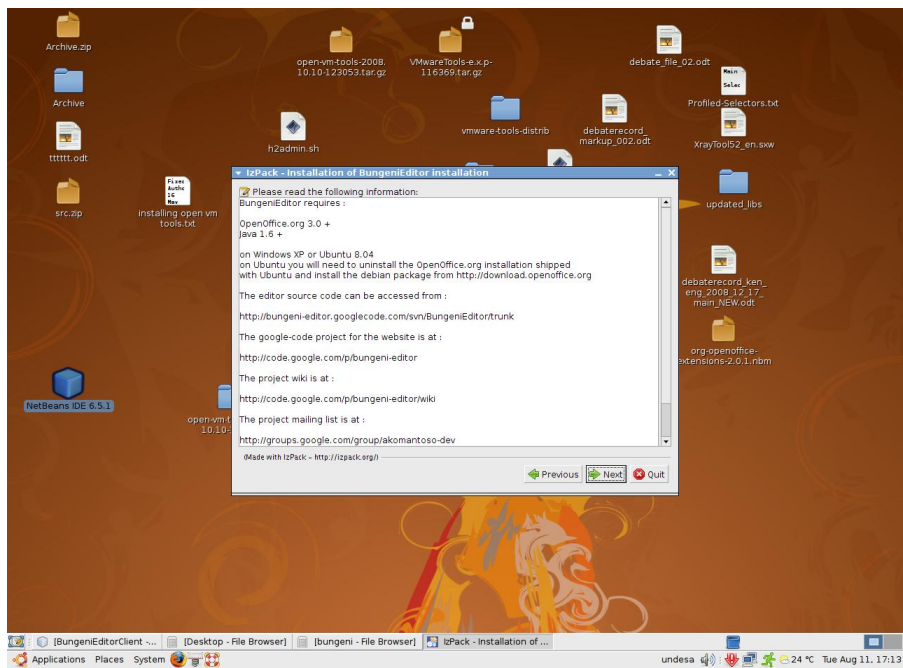
```
cd desktop-integration
sudo dpkg -i *.deb
```

Step 1 – Open the installer using the Sun Java 6 runtime

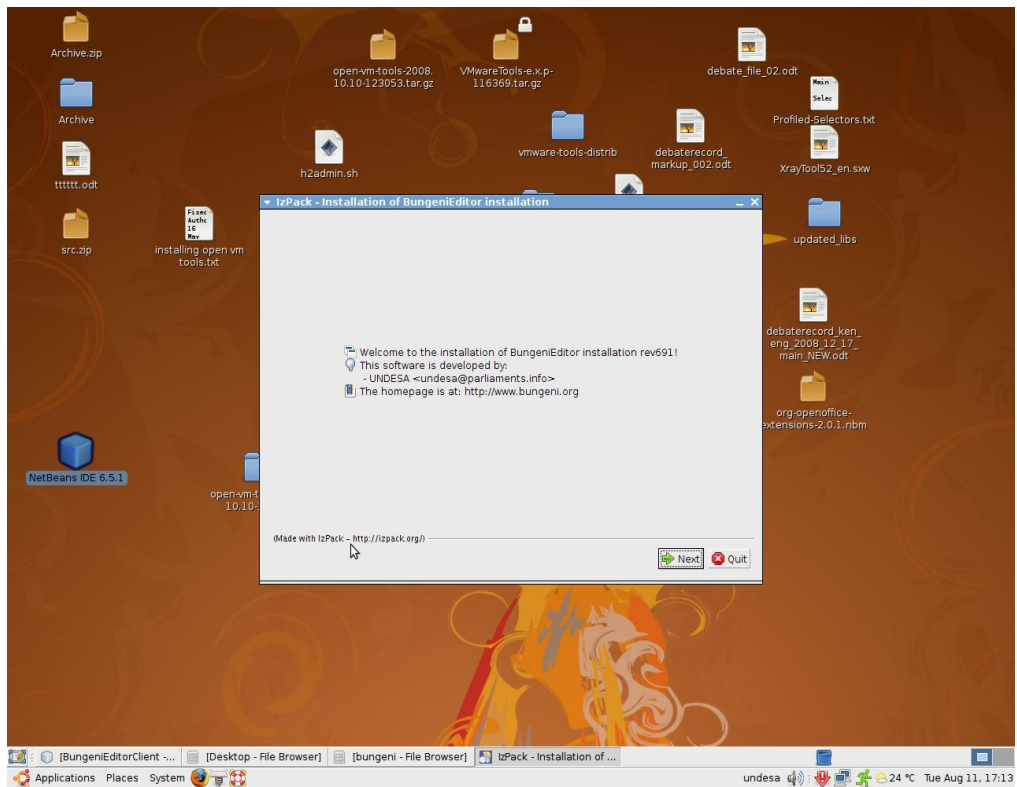


Right click on the installation jar file and select 'Open with Sun Java 6 runtime'

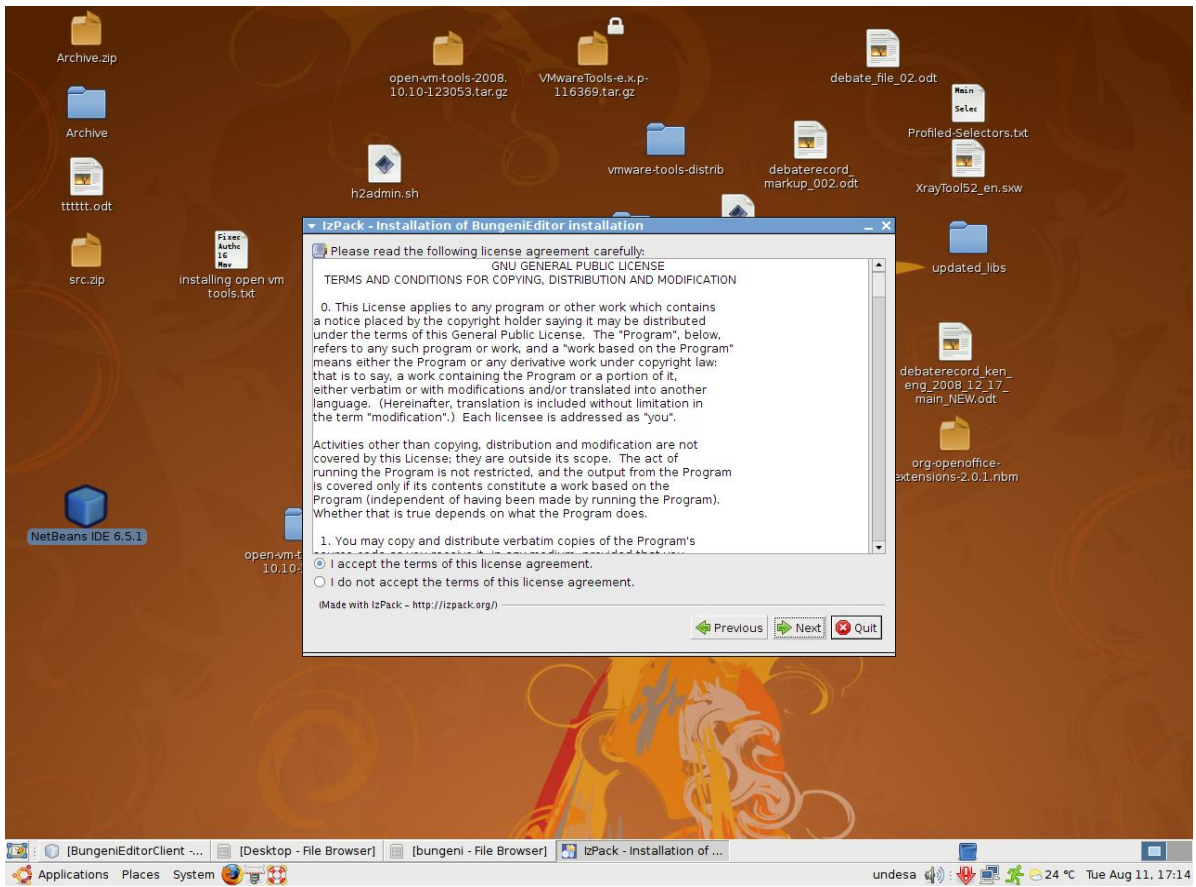
Step 2 – Provides a description of what will be installed and what is required for a successful installation.



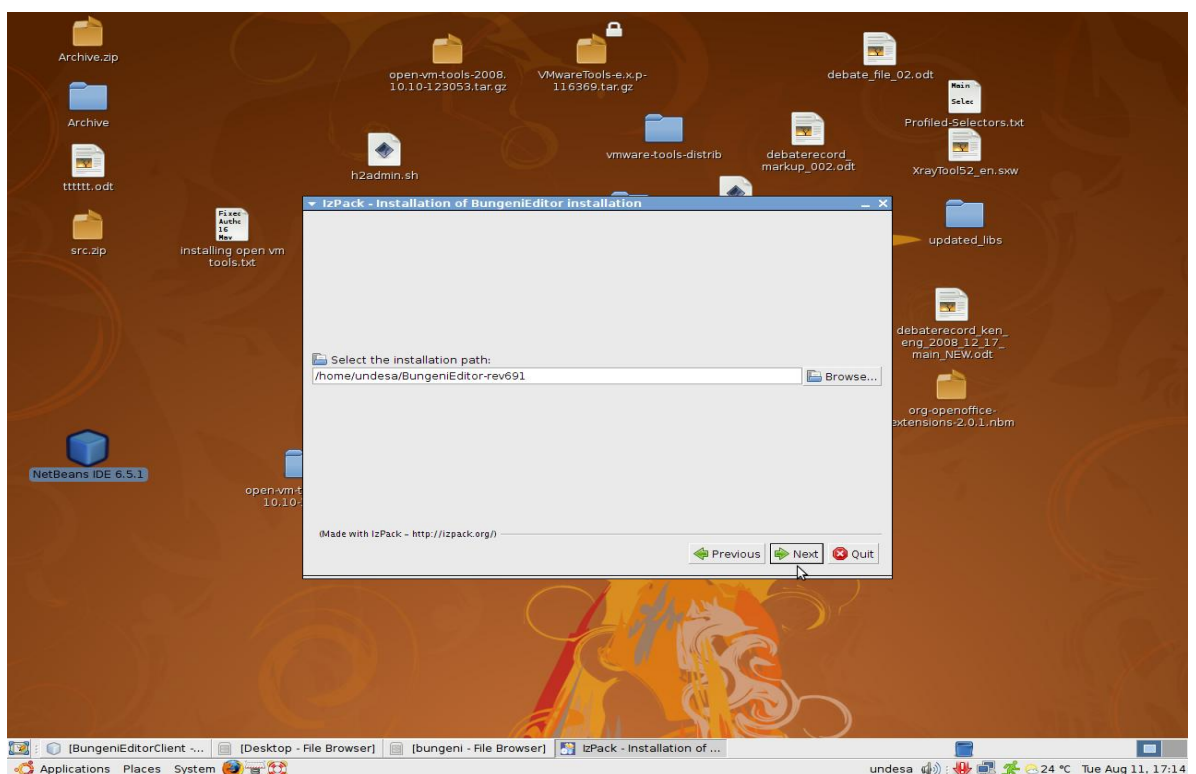
Step 3 – Information about the Project



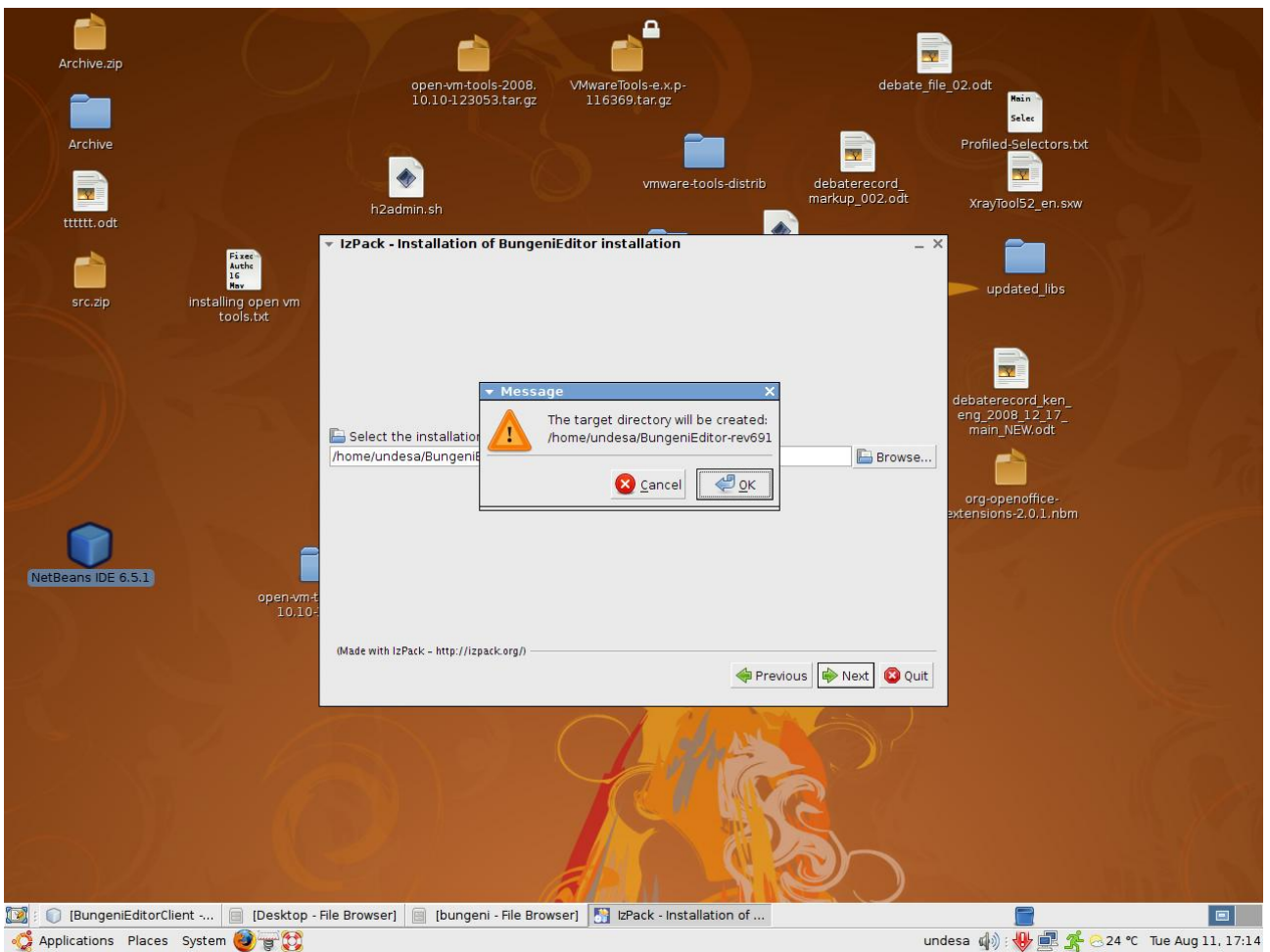
Step 4 – Licensing terms and conditions



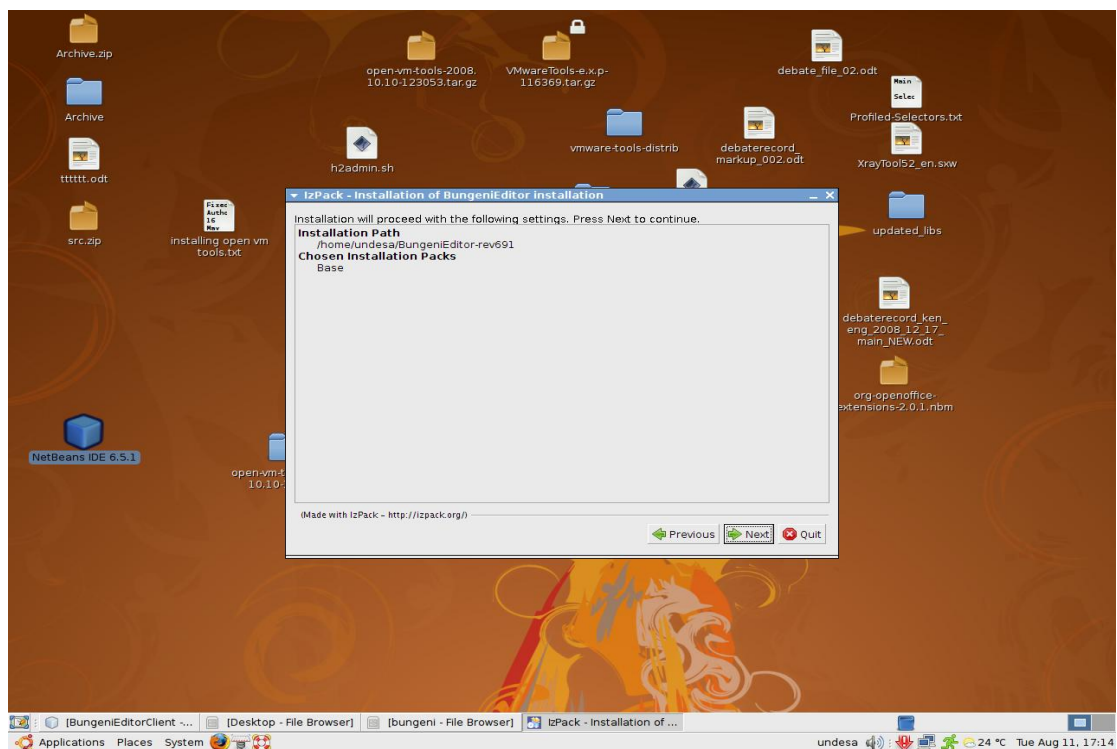
Step 5 – Select an installation directory



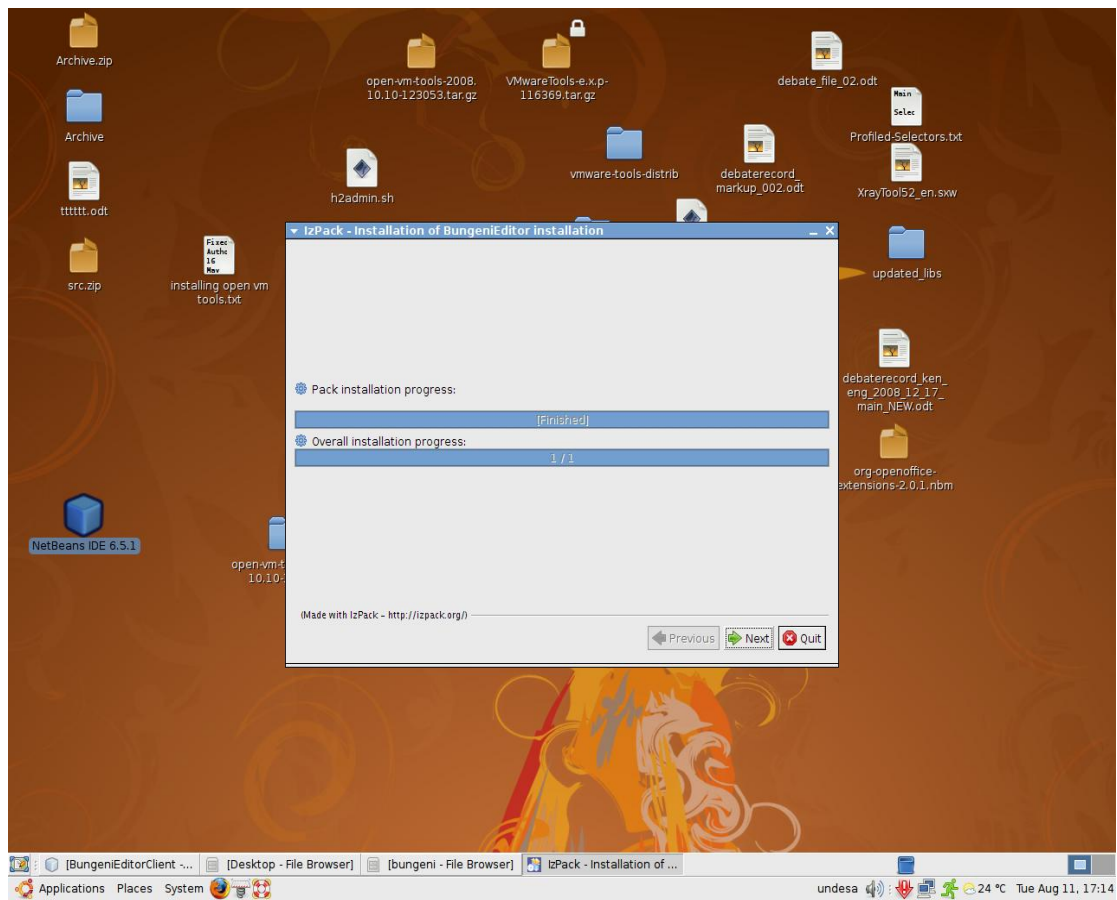
Step 6 – Agree to installation



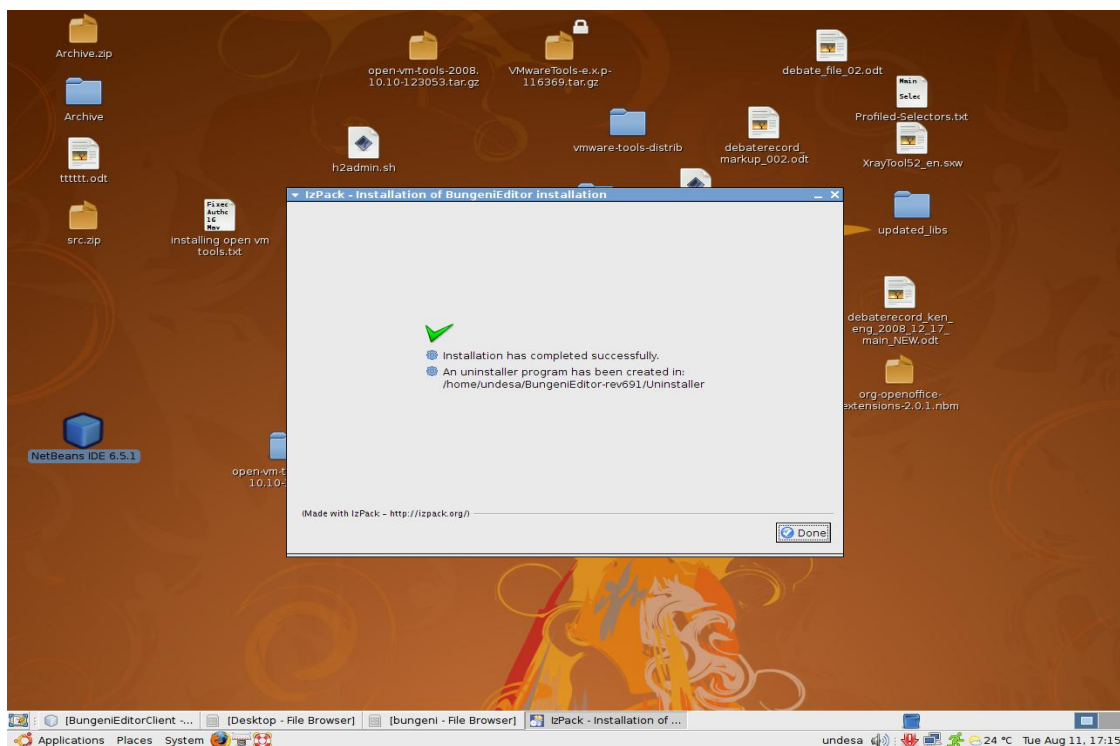
Step 7 - Informational screen – click next to install



Step 8 – Installation progress and completion



Step 9 – Installation completion screen



Installation Steps for Windows

Pre-requisites

The editor requires the following software as pre-requisites :

Sun Java 6 JRE

Download and install the Java 6 JRE for Windows from this link:

<http://www.java.com/en/download/manual.jsp>

The direct link to the installation packages is :

<http://javadl.sun.com/webapps/download/AutoDL?BundleId=33232>

OpenOffice.org 3.0.1

Download and install OpenOffice.org 3.0.1 (uninstall any previous versions of OpenOffice.org before doing so). From the below link :

http://openoffice.mirrors.tds.net/pub/openoffice/stable/3.0.1/OOo_3.0.1_Win32Intel_install_en-US.exe

Editor Installation

Installation steps are same as installing on ubuntu. Double click the installer and follow the graphical installer as for the Ubuntu installation described above.

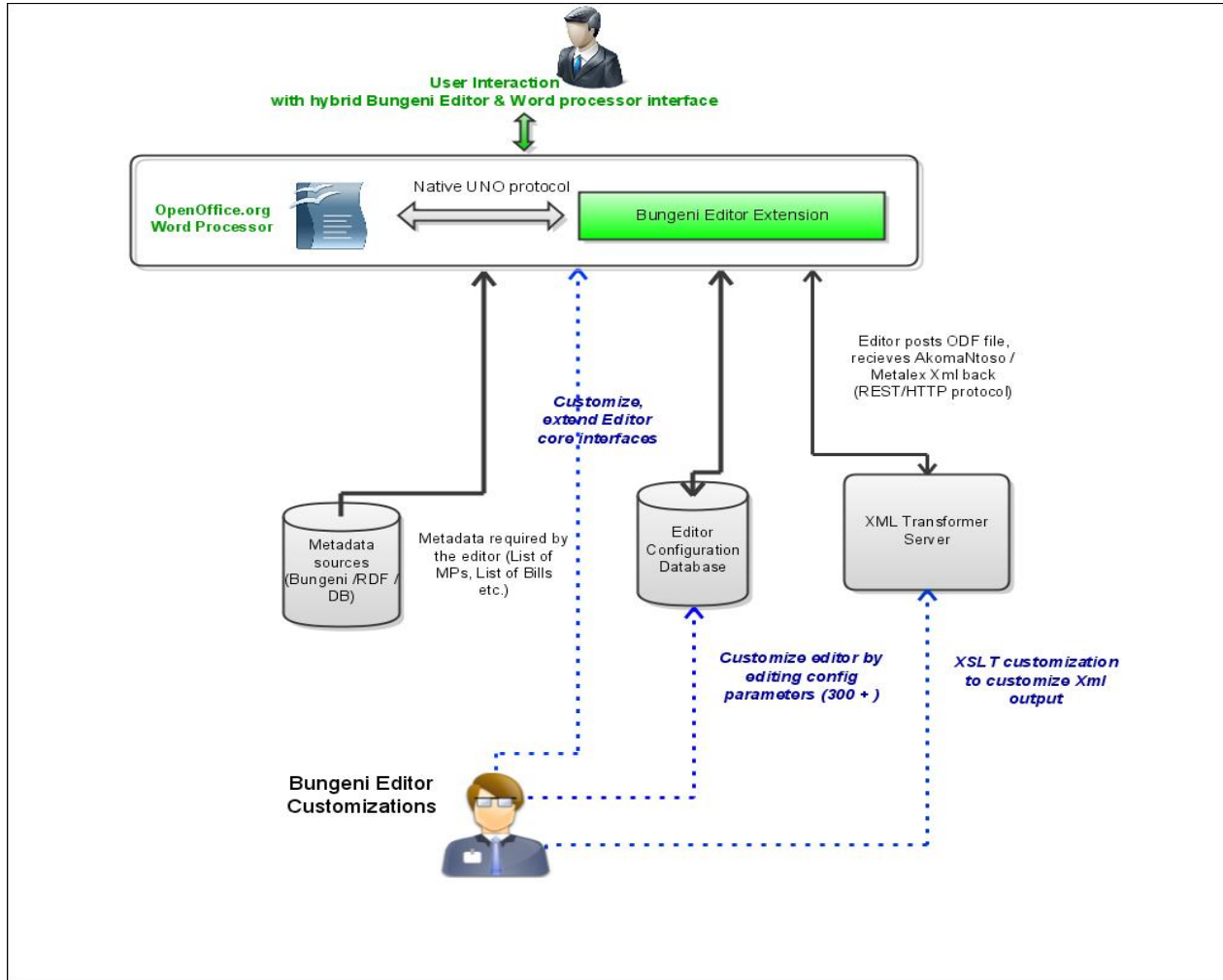
Caveats

Classloader errors

While launching the Editor the launch fails and you encounter 'ClassLoader errors'. This usually happens when :

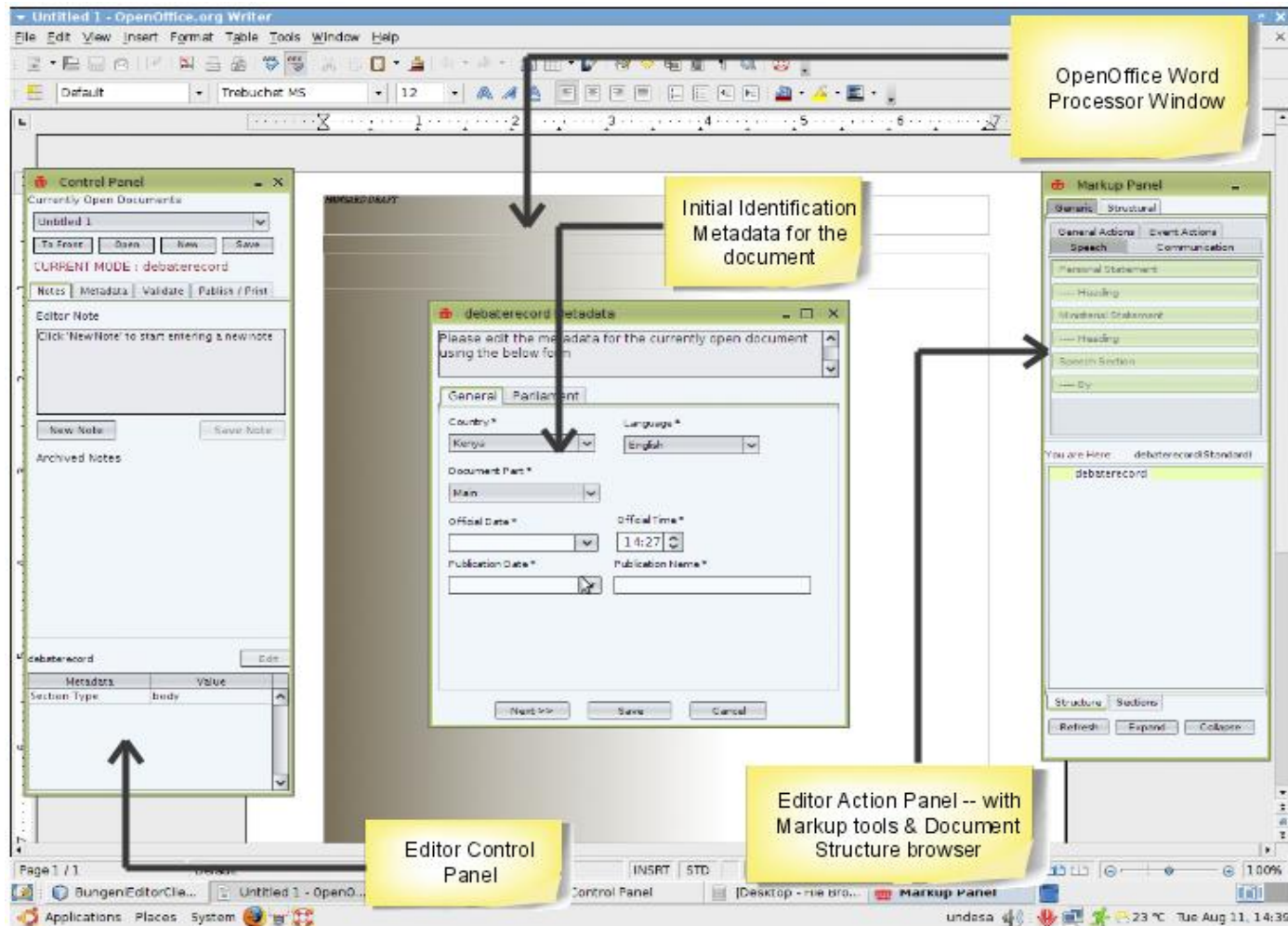
- 1) The editor is unable to detect an OpenOffice.org 3.0.1 installation – remedy : install OpenOffice.org 3.0.1
- 2) The editor detects an OpenOffice.org installation but it is not version 3.0.1 – remedy: uninstall the existing version and install OpenOffice.org. 3.0.1
- 3) On Ubuntu – if (1) or (2) occurs – you may be running 3.0.1 but the repackaged version distributed with Ubuntu. In that case follow the instructions here to install the correct openoffice version : link : OpenOffice.org 3.0.1 (installing openoffice 3.0.1 on Ubuntu)

Editor Architecture



The Editor User Interface

The Bungeni Editor user interface adds a couple of floating panes to the OpenOffice.org writer user interface. The middle area is the word processor window where the user edits the document. The left hand panel (“The Control Panel”) provides functionality to manage the workflow of legislative editing. The right hand panel (“The Markup Panel”) provides actions to markup the document and a structural view of the document being edited. Using the Editor



The Editing Process – Control Panel

The Editing process is managed primarily using the left hand “Control Panel”. The Control Panel uses a tabbed interface to group logically sequential steps of the editing process.

Step 1 Compose New / Edit Existing

Open existing documents or compose a new document. Switch between different documents; Add editor notes on status of the document corrections etc.

The screenshot shows the 'Control Panel' window with the following components:

- Currently Open Documents:** A dropdown menu showing 'ke_debaterrecord_2009-7-26_eng.o...'. Below it are buttons for 'To Front', 'Open', 'New', and 'Save'.
- CURRENT MODE:** 'debaterrecord'. Below this are buttons for 'Notes', 'Metadata', 'Validate', and 'Publish / Print'.
- Editor Note:** A text area containing the note: 'The speech has been incorrectly marked up within the point of order'. Below it are 'New Note' and 'Save Note' buttons.
- Archived Notes:** A list of notes with dates and times: '2009-07-22 09:20:37 [Ashok]', '2009-08-12 12:12:46 [Ashok]', and '2009-08-12 12:13:11 [Ashok]'.
- papers1:** A table with an 'Edit' button to its right.
- Metadata Table:** A table with two columns: 'Metadata' and 'Value'. It contains two rows: 'Is metadata editable' with value 'false', and 'Section Type' with value 'Papers Laid'.

Switch between different documents

Action buttons allow creating new document, open existing, save a document

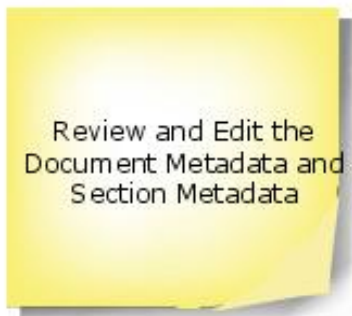
Users can add editorial notes which are saved within the document

Edit metadata of the active section

View the metadata of the currently active Section

Step 2 Metadata Panel

The Editor supports applying metadata either at the document level (i.e. The metadata is associated with the whole document) or with the main container element of the Editor the document section.



The screenshot shows the 'Control Panel' window with the 'Metadata' tab selected. It has two sub-tabs: 'Document' and 'Section'. The 'Document' sub-tab is active, displaying a table of document-level metadata. Below the table is an 'Edit Document Metadata' button. The 'Section' sub-tab is also visible, displaying a table of section-level metadata.

Name	Value
Work Date	2009-07-26
Country Code	ke
Manifestation D...	2009-07-21
Language ID	eng
Expression Date	2009-07-21
Official Date	2009-07-26
Parliament Ses...	12
Publication Name	debate-publication
Official Time	01:01
Parliament Sitting	3
Publication Date	2009-07-22
Author	
Document Part	main

papers1

Metadata	Value
Is metadata editable	false
Section Type	Papers Laid

The Section Sub-tab within metadata displays the section level metadata

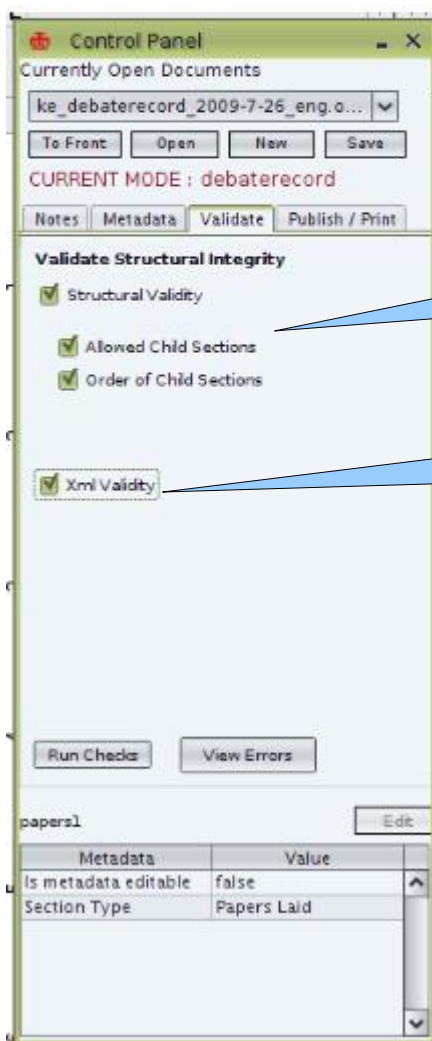
The Document Sub-tab within metadata displays the document level metadata

Clicking this button launches a form to edit Metadata

Step 3 – Validate and Check document

Check the Structural Integrity of the document & Validate XML

The Editor supports 2 levels of validation – the 'Structual Validation' is a semantic rule checker which allows creation of custom rules and calling them via a rules Xml config file. The Xml validation layer converts the document to AkomaNtoso and validates it against the AkomaNtoso schema.



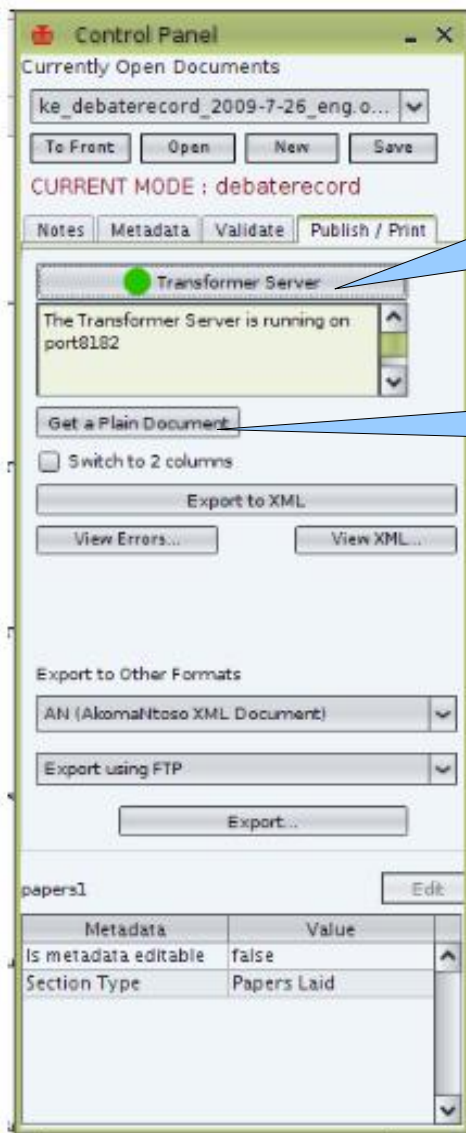
The structural checks – these can be selected and run optionally

Checking this option causes xml validity to be checked

Step 4 – Transform to different formats

Transform, Publish the Document to different formats

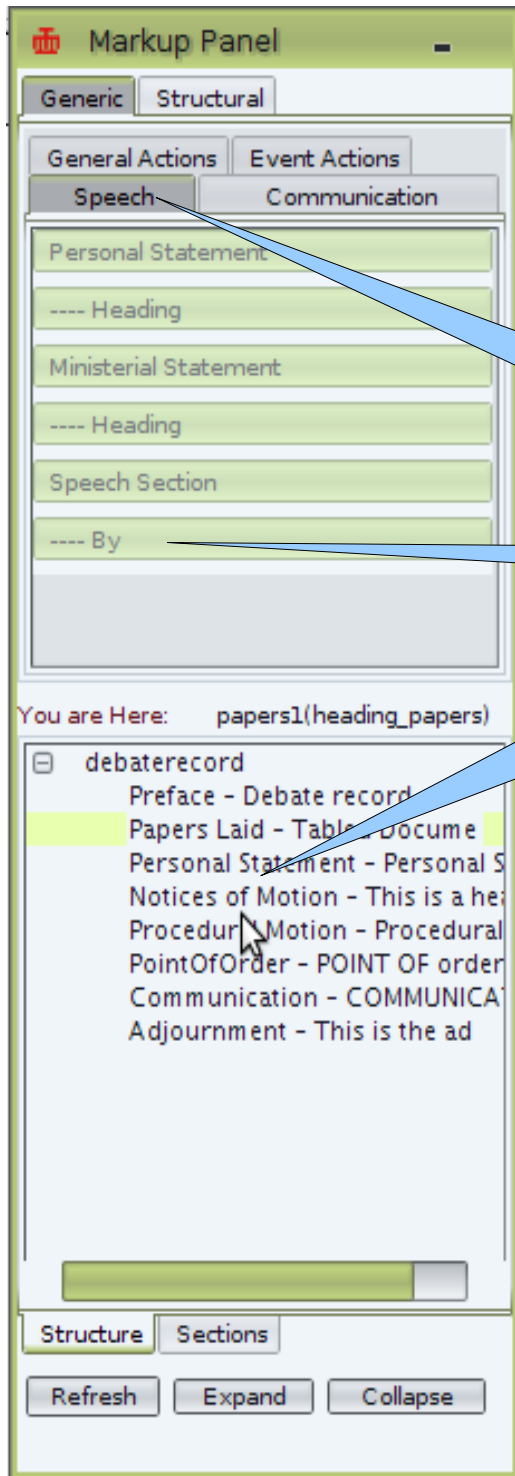
The Editor uses an XSLT based transformation engine.
The transformation from ODF to AkomaNtoso is a 2 step process – first the ODF document is converted to MetaLEX, and then the MetaLEX is transformed to AkomaNtoso



The green icon on the button indicates that the transformer server is up and running. The transformer server runs as a http service and integrates with the editor via a REST api

This converts the marked-up document to a plain document without any markup indentation or styling

The Editing Process – Markup Panel



On the right hand side of the word processor window is the 'Markup Panel' – this will be the commonly used window in the Editor as it contains the Markup actions.

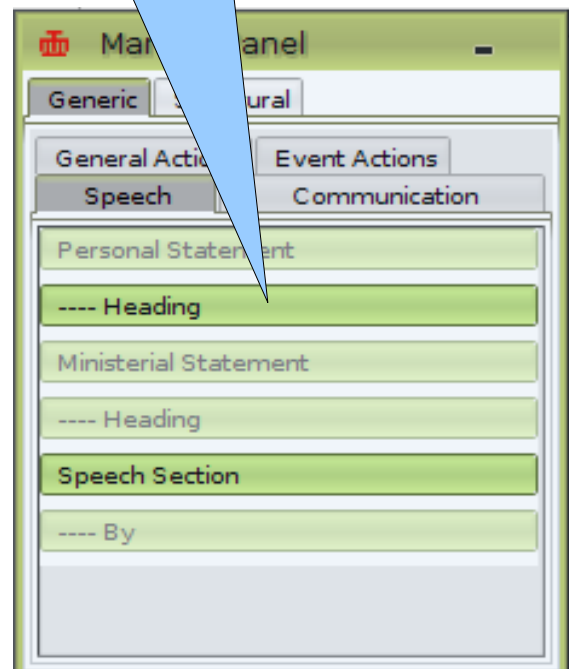
The Markup actions are grouped by Tabs and Sub-tabs – and the actions are enabled contextually. E.g. The heading action for the Speech element will be enabled only when the cursor highlights text within a Speech section (note that – in the context of the editor an AkomaNtoso Xml container element maps to a section).

Actions grouped by tabs

Individual Actions

Hierarchical structural view of document

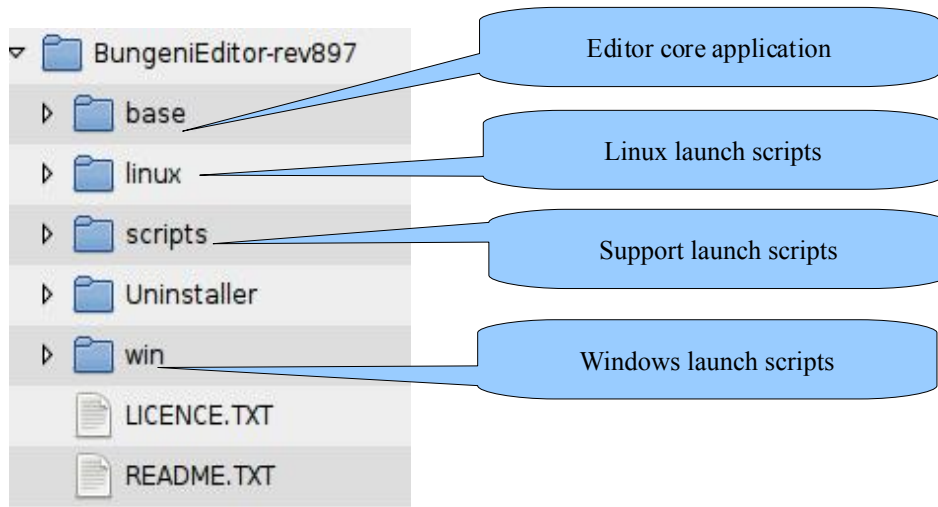
Markup panel with enabled ('Active') actions



Starting the Editor

The editor can be started using various helper scripts which are installed in the root directory of the editor installation folder.

Figure : Editor folder structure



The installation folder has various launch scripts which launch the bungeni editor in a specific contextual mode. The editor application can be launched with an active document type (For information about Document types see Document Types) - and subsequently for that session of editor usage the same document type remains active. The Editor supports editing and switching multiple documents at the same time – though they must all be of the same document type.

There are various launch scripts provided in the Editor's installation folder. Double clicking on a launch script launches the Editor in that mode. Launch scripts are grouped by operating system flavor – linux launch scripts are under the 'linux' folder , Windows launch scripts under the 'win' folder.

For e.g.

Clicking *run_debate_new.sh / run_debate_new.bat* - launches the editor with the active document type set as 'debaterecord', and creates a new empty document for the user to edit

Clicking *run_debate_edit.sh / run_debate_edit.bat* – launches the editor with the active document type set as 'debaterecord', and presents the user with file-open dialog to edit an existing file.

Clicking *settingsEditor.sh / settingsEditor.bat* – launches the UI for the settings / configuration database of the editor

Customizing and Extending the Editor

The Bungeni Editor allows customization at various levels.

The Editor architecture employs a vast number of configurable while allow customization of UI, Behavioral and Functional aspects of the Editor.

Editor Folder Structure

The folder structure of the Editor is as follows :

- lib – required libraries used by editor
- logs – editor outputs all debug otuput into a file in this folder
- plugins – this is a folder containing plugin extensions to the bungeni editor
- settings – this is a folder containing the settings / configuration database
- transformer – this is the folder containing the transformer server
- workspace – this is the folder containing the template for the document types and all the working documents

Document Types

Editor customization is based on the concept of a document type. The Bungeni Editor uses the OpenDocument format (ODF) to represent all legislative documents. ODF in turn allows setting of metadata at the document level and also for various artifacts like Images, Sections and Tables.

The Bungeni Editor identifies a ODF document as a legislative document based on certain custom properties it sets on the document. The most relevant of these properties is the 'Document Type' property – this allows the Editor to identify a document either as a bill, a debaterrecord, an act or a judgement.

The architecture of the Editor allows adding of new document types relatively easily – at present the Editor supports the following document types :

- DebateRecord
- Bill
- Judgement

Document Templates

The Bungeni Editor uses standard word processor templates to represent individual document types. This allows customization of styles and formatting per document type.

Settings Database

The Bungeni Editor uses an embedded database for its configuration registry. This configuration registry allows for granular setup and configuration of the Editor – and allows the editor to adapt to different legislative scenarios without re-writing or modifying source code.

The Settings configurations can be edited using a browser based UI. The Settings editor runs as a webserver on the local editor installation and can be accessed using any web-browser on the localhost. The settings editor runs on port 8082 by default and can be accessed by browsing to <http://localhost:8082>.

The Settings editor provides a hierarchical view of various configuration options for the Editor. This is described in detail in the “Advanced Configuration section”

Fig : Settings Database configuration editor

The screenshot shows a web browser window displaying the Settings Database configuration editor. The browser's address bar shows the URL `http://localhost:8082/frame.jsp?sessionId=07a7b971f85103ebbb8c3ae31816696b`. The interface includes a menu bar (File, Edit, View, History, Bookmarks, Tools, Help), a toolbar with navigation and execution buttons, and a sidebar with a tree view of configuration categories. The main area displays a table of configuration properties for the 'GENERAL_EDITOR_PROPERTIES' table.

Action	PROPERTY_NAME	PROPERTY_VALUE
	structuralRulesRootPath	settings/structural_rules
	pluginsPath	plugins
	activeProfile	withBackend
<input checked="" type="checkbox"/>	attributeDateFormat	<input type="text" value="yyyy-MM-dd"/>
<input type="checkbox"/>	defaultWorkURI	CountryCode.FullDate.PartName
	defaultHierarchyView	VIEW_PRETTY_SECTIONS
	registryJDBCdriver	org.h2.Driver
	registryDB	registry.db
	registryJDBCdriverPrefix	jdbc:h2:
	logoPath	settings/logos
	activeDocumentMode	debaterecord
	rootdebaterecord	debaterecord
	localRegistry	yes
	registryUser	sa
	registryPassword	
	localRegistryFolder	registry

UI Theme

The UI of the editor is fully “skinnable”. The Bungeni Editor uses a skinning engine called “Substance” - there are various default skins provided, and can be switched via a configuration file. Additional themes can be created using the Java Look-and-Feel API.

The active them can be changed by editing the `bungenitheme.properties` configuration file in the 'settings' folder and changing the 'DefaultTheme' property ::

```
#####The following themes are available #####
# org.bungeni.editor.themes.BusinessBlackSteelLAF
# org.bungeni.editor.themes.BusinessBlueSteelLAF
# org.bungeni.editor.themes.BusinessLAF
# org.bungeni.editor.themes.CafeCremeLAF
# org.bungeni.editor.themes.CremeLAF
# org.bungeni.editor.themes.NebulaLAF
# org.bungeni.editor.themes.NebulaBrickWallLAF
# org.bungeni.editor.themes.SaharaLAF
# org.bungeni.editor.themes.MotifLAF
# org.bungeni.editor.themes.ModerateLAF
# org.bungeni.editor.themes.GtkLAF
# org.bungeni.editor.themes.OfficeBlue2007LAF
# org.bungeni.editor.themes.AutumnLAF
# org.bungeni.editor.themes.MagmaLAF
#
# You can change the default theme by setting it to
# one of the themes specified above
#####
DefaultTheme = org.bungeni.editor.themes.SaharaLAF
```

See <http://code.google.com/p/bungeni-editor/wiki/EditorTheming>

i18n – Internationalization & Localization

The Bungeni Editor is an i18n compliant application. String messages and UI labels are all loaded from multilingual resource bundles.

Further – dynamically displayed UI aspects (like displaying structural information about a document) are also i18n compliant.

For instance – metadata name and section type name aliases are all loaded from message bundles in the settings/bundles folder.

Editor Markup Actions

The Bungeni Editor provides a way to create “Markup” actions. These are user actions that directly change the document – and also apply semantic meaning to the content at the same time.

A good example of a “Markup action” is the process of marking up a “Preface” for a debater record – the user selects the piece of text to be identified as a “Preface” -- the corresponding action in the “Editor Action Panel” (see Error: Reference source not found) gets enabled. When the user moves the mouse over to the “Action Panel” and clicks “Markup Preface” -- the highlighted text in the document is “marked up” as a “Preface”. The markup action embeds information in the highlighted text in the ODT document to identify that fragment as a “Preface”.

Markup actions are defined using a custom XML format. The Editor allows associating a set of markup actions (grouped in a Toolbar action Xml config file) to a document type. This is done in the TOOLBAR_XML_CONFIG table in the Settings Database. Sample settings for the TOOLBAR_XML_CONFIG table are shown below :

Document types are mapped to individual action configuration files. The path is relative to the installation folder.

DOC_TYPE	TOOLBAR_XML
bill	settings/toolbar_bill.xml
debaterecord	settings/toolbar_debate_b.xml
judgement	settings/toolbar_judgement.xml

The complete documentation for writing action configuration files can be found on the Bungeni Editor wiki on this link See :

<http://code.google.com/p/bungeni-editor/wiki/ImplementingACompleteAction>

The action Xml files use a syntax based condition processor to evaluate if an action is applicable in a particular context of the current fragment being marked up in the document. The condition processor has been documented here :

<http://code.google.com/p/bungeni-editor/wiki/ConditionProcessorsInBungeniEditor>

Editor Plugins

Introduction

The Bungeni Editor supports adding of functionality via plugins. As it is likely that plugins may use the same libraries used by the editor or in some instances different to the version of libraries used by the editor -- they are loaded and instantiated using a customized class-loader.

A post-delegation classloader is used to load the plugin and its required libraries. In most cases, if the plugin uses the same version of the library as the editor it is safe to not include the same library for the plugin as the post-delegated classloader uses the instance of the library from the parent (i.e. the Editor's) classloader.

Plugin Definition

Plugins are described in xml configuration files within the 'plugins' subfolder of the editor installation. The following is an example plugin config file :

```
<plugin name="ConvertToPlain" base="convert_to_plain" >
  <main
class="org.bungeni.plugins.convertsection.BungeniSectionConvertToPlain">BungeniSectionPlain.jar</main>
  <required>
    <lib>bungeniodfdom.jar</lib>
    <lib>odfdom.jar</lib>
    <lib>log4j.jar</lib>
  </required>
</plugin>
```

The description of the individual components of the file is provided below :

- `<plugin name="ConvertToPlain" base="convert_to_plain" >` - *'name' is the descriptive name of the plugin used by the system ; 'base' is the name of the folder containing the plugin jar files.*
- `<main class="org.bungeni.plugins.convertsection.BungeniSectionConvertToPlain">`

ain" >BungeniSectionPlain.jar</main> - 'class' is the entry point class for the plugin ; and the <main> element stores the name of the plugin jar file.

- <required><lib>.....</lib>... - this contains the name of the required libraries for the plugin and which reside in the base folder. If a library from the parent classloader can be used by the plugin, do not include it here.

Plugin configuration files use the following naming scheme : ***plugin_anytexthere.xml***

Required Interfaces

The plugin is required to implement the [IEditorPlugin interface](#).
The plugin methods are invoked by the editor using reflection.

Building the Editor from Source

Pre-requisites

- Sun Java 6 JDK
- Ant
- OpenOffice.org 3.0.1

Building from source

First SVN checkout the source.

Create a folder for the editor source, and then check the source out into it :

```
svn co http://bundeni-editor.googlecode.com/svn/BungeniEditor/trunk/BungeniEditorClient
```

We use ant for building the project. The build script takes a parameter specifying the path to the openoffice installation :

```
ant -buildfile build-cmdline.xml -Dopenoffice.org.root=/opt/openoffice.org/basis3.0 jar
```

This creates the editor jar file 'BungeniEditorClient.jar' under the dist folder and the required resource folders