BASIS TECHNOLOGY

Rosette LINGUISTICS PLATFORM

For Lucene

Application Developer's Guide

Release 6.0.0

September 2008

Table of Contents

1. Introduction

The **RLP For Lucene** SDK package integrates **RLP** with **Lucene** and **Solr**. You can use this package to enhance applications that index and search English and Japanese documents, as well as documents in any of the other languages for which **RLP** provides support.

For the API documentation that accompanies this code, consult the Java API Reference [api-reference/index.html].

The Lucene Package. The `com.basistech.rlp.lucene` package provides a tokenizer (`RLPTokenizer`), three Lucene analyzers that use this tokenizer (`RLPAnalyzer`, `RLPJaAnalyzer`, and `RLPEnAnalyzer`), and a Lucene filter (`RLPPOSFilter`) for identifying tokens of interest based on their part of speech.

`RLPTokenizer` is designed to be as flexible as possible, and it can be used for any of the languages that **RLP** supports.

`RLPAnalyzer` is language neutral, but you need to provide appropriate parameters (such as language and RLP context configuration) for a given language of interest [10] .

`RLPEnAnalyzer` and`RLPJaAnalyzer` are for English and Japanese respectively. They use `RLPAnalyzer` with a set of configuration data for each language. You can fine-tune the data to meet your needs, or you can use the source code of these Analyzers as a starting point for creating more specialized analyzers. See Creating Your Own Analyzers [10] .

`RLPPOSFilter` enables you to filter out tokens of unwanted parts of speech. It works with a `TokenStream` generated by `RLPTokenizer`, and a file containing the list of POS tags you want to accept. POS tags vary depending on the language, so you need a file for each language of interest. The SDK includes filter files for English and Japanese, which you can modify if you wish. For the lists of available POS tags for each language, refer to "Appendix: Part-of-Speech Tags" in the *RLP Application Developer's Guide*.

The Demo Package. The `org.apache.lucene.demo` package contains a version of the standard **Lucene** demo programs, modified to use the RLP analyzers. See Running the Lucene Command-Line-Interface Demos [4] .

The Solr Package. The `com.basistech.rlp.solr` package provides factory classes for `RLPTokenizer` and `RLPPOSFilter` classes: `RLPTokenizerFactory` and `RLPPOSFilterFactory`.

To integrate **RLP** with **Solr**, the only file you need to modify is the **Solr** configuration file: **schema.xml**. This file is distributed with **Solr**.

2. What you Need

- **RLP** version 6.0 or later with a license to run BL1 for the English Analyzer, and JLA and RCLU for the Japanese Analyzer. The English Analyzer uses BL1 to perform base linguistic analysis of English text. The Japanese Analyzer uses JLA to perform base linguistic analysis of Japanese text and RCLU to normalize character width variations (Unicode Normalization Form KC).
- **Lucene** 2.3.1 or later (Java release) or **Solr** 1.3. (which includes Lucene). For **Lucene**, see http://lucene.apache.org/java/. For **Solr**, see http://lucene.apache.org/solr/.
- **Java Standard Edition JDK** 1.5 or later. See http://java.sun.com.

- **Ant** 1.65 or later. See http://ant.apache.org.
- A web application server if your application is a web application. *Note:* **Solr** comes with a web application server to run the demo web application.
- The compressed **RLP For Lucene** SDK package file.

  Basis distributes this package in two forms: a **.zip** file (for Windows) and a **.tar.gz** file (for Unix).

  SDK Package File Names:
  - **rlplucene-6.0.0-sdk-win.zip**
  - **rlplucene-6.0.0-sdk-unix.tar.gz**
- The compressed **RLP For Lucene** documentation package file.

  The documentation package file includes this guide, release notes, and HTML API reference for the Java API. Basis distributes this package in two forms: a **.zip** file (for Windows) and a **.tar.gz** file (for Unix).

  SDK Documentation File Names:
  - **rlplucene-6.0.0-doc-win.zip**
  - **rlplucene-6.0.0-doc-unix.tar.gz**

3. Installing RLP for Lucene

Where? For convenience, we recommend you install **RLP For Lucene** in the same directory where you installed **RLP**. This document refers to the directory where **RLP** is installed as *BT_ROOT*. You do not have to install **RLP For Lucene** in *BT_ROOT*, but if you do, the **RLP For Lucene** Java API documentation provides useful links to the **RLP** Java API documentation. If you install **RLP For Lucene** to another location, these links will not work.

Installation Procedure

1. Expand the **RLP For Lucene** SDK package to *BT_ROOT*.

   Windows example:
   a. In **Explorer**, double click the package file.
   b. In the Compressed Folder window that appears, use the **Extraction Wizard** to extract the package to *BT_ROOT* (for example, **C:\BasisTech**).

   Unix example (Bash):
   a. Put the file you have downloaded in *BT_ROOT* (for example, **/usr/local/BasisTech**).
   b. Extract the package in that directory.

   ```
   mv rlplucene-6.0.0-sdk-unix.tar.gz /usr/local/BasisTech
   cd /usr/local/BasisTech
   gunzip rlplucene-6.0.0-sdk-unix.tar.g
   tar -xf rlplucene-6.0.0-sdk-unix.tar
   ```

2. Expand the **RLP For Lucene** Documentation package to *BT_ROOT*, the same directory where you installed the SDK.

   This guide (**RLP-For-Lucene.pdf** and the release notes (**rlplucene-6.0.0-ReadMe.html**) are placed in the **rlplucene/doc** subdirectory.

The Java API Reference is placed in the **rlplucene/doc/api-reference** subdirectory. The Java API reference also includes links to the **RLP** Java API documentation, which should be installed in ***BT_ROOT*/rlp/doc/api-reference/java-reference**.

4. Setting Environment Variables

For brevity, many of the instructions in this document assume that you have set the environment variables described in this section. The only environment variable you are required to modify (on Unix only) is `LD_LIBRARY_PATH` or its equivalent when you run a Java program that uses RLP For Lucene. If you prefer not setting the other environment variables, you must substitute the appropriate values when you use these instructions.

In Windows, you can use the **Control Panel** (select **System→Advanced→Environment Variables**) or the `SET` command. In Unix, you can use the Bash `export` command.

Variable	Details
`BT_ROOT`	Set `BT_ROOT` to refer to the directory in which you installed the RLP SDK. It is the parent of the **rlp** and **build_system** directories. The Ant build script uses `BT_ROOT` to set the `bt.root` system property when it launches the Java virtual machine.
`BT_BUILD`	Set `BT_BUILD` to refer to the platform identifier embedded in your RLP SDK package file, such as `ia32-w32-msvc71` or `ia32-glibc22-gcc-32`. A directory with this name also appears in the ***$BT_ROOT*/rlp/lib** directory. The Ant build script uses `BT_BUILD` in conjunction with `BT_ROOT` to find the Java library.
`JAVA_HOME`	Set `JAVA_HOME` to refer to the directory where you installed the Java SDK 1.5 or higher.
`SOLR_HOME`	If you are using Solr, set `SOLR_HOME` to refer to the directory where you installed Solr 1.3.
`PATH`	Set `PATH` to include the Java and Ant bin directories.
(Unix only) `LD_LIBRARY_PATH`	Or its equivalent for your Unix platform. On Unix platforms, you must set this variable to include **$BT_ROOT/rlp/lib/$BT_BUILD**. For example: `export LD_LIBRARY_PATH=$BT_ROOT/rlp/lib/$BT_BUILD:$LD_LIBRARY_PATH`

5. Running the Lucene Command-Line-Interface Demos

The `org.apache.lucene.demo` package contains a version of the standard **Lucene** demo programs, modified to use the RLP analyzers.

Ant 1.6.5 or higher is required to run these demos.

Environment Variables

To use the Ant build script as described below, you must set several environment variables [4] , including `LD_LIBRARY_PATH` or its equivalent on Unix platforms.

5.1. Indexing Documents

From the Windows Command Prompt:

```
ant -Dbt.root=%BT_ROOT% -Dbt.arch=%BT_BUILD% -Ddemo.lang=en runindexdemo
```

From a Unix shell, such as Bash:

```
ant -Dbt.root=$BT_ROOT -Dbt.arch=$BT_BUILD -Ddemo.lang=en runindexdemo
```

runs the **Lucene** indexing demo program that has been modified to use the `RLPEnAnalyzer` default constructor indirectly via the `RLPAnalyzerDispatcher` helper class, to index and search the English plain text files found in the ***BT_ROOT*/rlp/samples/lucene/sampledocs/en** subdirectory. The **Lucene** index is created in the **index** directory.

5.2. Searching Documents

From the Windows Command Prompt:

```
ant -Dbt.root=%BT_ROOT% -Dbt.arch=%BT_BUILD% -Ddemo.lang=en runsearchdemo [-Dqueries=queryFile]
```

From a Unix shell, such as Bash:

```
ant -Dbt.root=$BT_ROOT -Dbt.arch=$BT_BUILD -Ddemo.lang=en runsearchdemo  [-Dqueries=queryFile]
```

runs the search demo program, where the optional *`queryFile`* specifies a UTF-8 file with a list of strings to search for (one per line). To use the query file that we provide, use `-Dqueries=queries/en.txt`.

If you do not include `-Dqueries=`*`queryFile`* , the demo prompts you to enter a search string. For example, type "basis" and press **Enter**; the demo should report 5 matches. If you type "say" and press **Enter**, the list of matching files should include **sampledocs/en/lucene-070327.txt**. This file, however, does not contain the word "say". It only has the verb's past form "said". RLPEnAnalyzer automatically added the base form of the word to the index and the file was found. Try other strings. When done, press the **Enter** key.

5.3. Removing the Index

```
ant cleandemoindex
```

removes the index file directory. This is necessary before running the indexing program again.

5.4. Running the Demo with Japanese Text

To run the demo for Japanese, replace `-Ddemo.lang=en` with `-Ddemo.lang=ja`. The demo will index and search the plain text files in UTF-8 encoding found in the ***BT_ROOT*/rlp/samples/lucene/sampledocs/ja** subdirectory.

For search strings, we recommend you include `-Dqueries=`*`queryFile`* where *`queryFile`* is a UTF-8 file with a list of Japanese query strings (one per line).

To use the query file that we provide, use `-Dqueries=queries/ja.txt`.

If you do not supply the queries file, you must have set your console to handle Japanese text.

If you do not include `-Dqueries=`*`queryFile`* , the demo prompts you to enter a search string. Assuming you have set your console to handle Japanese text, you can use a string like "新聞" (newspaper) and press **Enter**. Try other strings. When done, press the **Enter** key. [1]

[1] If your system is not set up to use the Japanese code page, you cannot enter Japanese queries. If you use a UTF-8 queries file, the results displayed on the console will include question marks in the place of each query string. You can use the queries file to identify each query string.

6. How To Use

6.1. Using with Lucene

Procedure

1. Write or modify your search applications to use `RLPAnalyzer`, `RLPEnAnalyzer`, or `RLPJaAnalyzer` where an analyzer is expected.

2. For web applications, you must copy **btrlp.jar** and **btutil.jar** from ***BT_ROOT*/rlp/lib/*BT_BUILD*** to the application server's common library directory (not the web application-specific libary directory). If you are using **Tomcat** 5.x, the common library directory is ***CATALINA_HOME*/shared/lib**, where *CATALINA_HOME* is the **Tomcat** installation directory.

   Also, copy the integration sample code jar file, **btrlplucene.jar**, from ***BT_ROOT*/rlp/lib/*BT_BUILD*** into either the application-specific library directory (**WEB-INF/lib**) or the server's common library directory.

3. Set the system property `bt.root` to point to *BT_ROOT*.

   If you are using **Tomcat** 5.x on Windows, you would specify this by right-clicking on the **Tomcat** icon, selecting Configure, choosing Java tab, and adding: -Dbt.root=*BT_ROOT* on each line in the text box named Java Options (replace *BT_ROOT* with the actual path).

   If you run your application from the java command line, add the command line option `-Dbt.root=%BT_ROOT%` (Windows) or `-Dbt.root=$BT_ROOT` (Unix). The classpath must be set to include the jar files mentioned above.

6.2. Using with Solr

The following instructions are to run **Solr** with the RLPTokenizer for Japanese or English analysis.

We assume you have downloaded and installed **Solr** 1.3, and that you have set `SOLR_HOME` to the directory where you have installed Solr 1.3.

We also assume that:

- You are adding a new field type called `text_ja` (for Japanese) or `text_en` (for English).
- You are using the RLP tokenizer for both indexing and queries.
- For indexing, you are using RLPPOSFilter with the default set of the POS tags. You are not using RLPPOSFilter for queries, because short query strings to not provide enough linguistic context to accurately identify part-of-speech tags.
- You continue to use `LowerCaseFilter` for case-insensitive queries.

Procedure

1. Copy the following files from **RLP** and **RLP For Lucene** to **Solr** as directed below:

File(s) to copy	Destination Directory
***BT_ROOT*/rlp/lib/*BT_BUILD*/btrlp.jar**	***SOLR_HOME*/example/lib/ext**
***BT_ROOT*/rlp/lib/*BT_BUILD*/btutil.jar**	
rlplucene/lib/btrlpextra.jar	
rlplucene/lib/slf4j-api-1.5.2.jar	
rlplucene/lib/slf4j-jdk14-1.5.2.jar	
rlplucene/lib/btrlplucene.jar [a]	***SOLR_HOME*/example/solr/lib**
rlplucene/conf/*	***SOLR_HOME*/example/solr/conf**

[a] If you have modified and rebuilt rlplucene.jar [11], copy **rlplucene/build/rlplucene.jar**.

In the Windows Command Prompt:
Replace *RLPLUCENE* with the path to the **rlplucene** directory.

```
cd RLPLUCENE
mkdir %SOLR_HOME%\example\lib\ext
mkdir %SOLR_HOME%\example\solr\lib
copy %BT_ROOT%\rlp\lib\%BT_BUILD%\btrlp.jar %SOLR_HOME%\example\lib\ext
copy %BT_ROOT%\rlp\lib\%BT_BUILD%\btutil.jar %SOLR_HOME%\example\lib\ext
copy lib\btrlpextra.jar %SOLR_HOME%\example\lib\ext
copy lib\slf4j-api-1.5.2.jar %SOLR_HOME%\example\lib\ext
copy lib\slf4j-jdk14-1.5.2.jar %SOLR_HOME%\example\lib\ext
copy lib\btrlplucene.jar %SOLR_HOME%\example\solr\lib
copy conf\* %SOLR_HOME%\example\solr\conf
```

In the Unix shell, such as Bash:
Replace *RLPLUCENE* with the path to the **rlplucene** directory.

```
cd RLPLUCENE
mkdir -p $SOLR_HOME/example/lib/ext
mkdir -p $SOLR_HOME/example/solr/lib
cp $BT_ROOT/rlp/lib/$BT_BUILD/btrlp.jar $SOLR_HOME/example/lib/ext
cp $BT_ROOT/rlp/lib/$BT_BUILD/btutil.jar $SOLR_HOME/example/lib/ext
cp lib/btrlpextra.jar $SOLR_HOME/example/lib/ext
cp lib/slf4j-api-1.5.2.jar $SOLR_HOME/example/lib/ext
cp lib/slf4j-jdk14-1.5.2.jar $SOLR_HOME/example/lib/ext
cp lib/btrlplucene.jar $SOLR_HOME/example/solr/lib
cp conf/* $SOLR_HOME/example/solr/conf
```

2. (Unix only). If you have not already done so, set LD_LIBRARY_PATH (or the equivalent environment variable for your Unix platform) to include the RLP shared libraries directory:

```
export LD_LIBRARY_PATH=$BT_ROOT/rlp/lib/$BT_BUILD:$LD_LIBRARY_PATH
```

3. Edit *SOLR_HOME*/example/solr/conf/schema.xml to include the following entries in the <types> element, replacing *SOLR_HOME* with the actual absolute path.

   For Japanese:

```
<fieldtype name="text_ja" class="solr.TextField" positionIncrementGap="100">
 <analyzer type="index">
  <tokenizer class="com.basistech.rlp.solr.RLPTokenizerFactory"
              rlpContext="SOLR_HOME/example/solr/conf/rlp-context-jla.xml"
```

```
                    lang="ja"
                    postStem="true"
                    postCompoundComponents="true"/>
    <filter class="com.basistech.rlp.solr.RLPPOSFilterFactory" pos="pos-ja.txt"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="com.basistech.rlp.solr.RLPTokenizerFactory"
                    rlpContext="SOLR_HOME/example/solr/conf/rlp-context-jla.xml"
                    lang="ja"
                    postCompoundComponents="true"
                    postPartOfSpeech="false"/>
    <filter class="solr.LowerCaseFilterFactory"/>
   </analyzer>
</fieldtype>
```

For English:

```
<fieldtype name="text_en" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="com.basistech.rlp.solr.RLPTokenizerFactory"
                    rlpContext="SOLR_HOME/example/solr/conf/rlp-context-bl1.xml"
                    lang="en"
                    postStem="true"
                    postCompoundComponents="true"/>
    <filter class="com.basistech.rlp.solr.RLPPOSFilterFactory" pos="pos-en.txt"/>
    <filter class="solr.LowerCaseFilterFactory"/>

  </analyzer>
  <analyzer type="query">
    <tokenizer class="com.basistech.rlp.solr.RLPTokenizerFactory"
                    rlpContext="SOLR_HOME/example/solr/conf/rlp-context-bl1.xml"
                    lang="en"
                    postCompoundComponents="true"
                    postPartOfSpeech="false"/>
    <filter class="solr.LowerCaseFilterFactory"/>
   </analyzer>
</fieldtype>
```

4. Run **start.jar** with the `bt.root` system property set inside the example subdirectory.

   In the Windows Command Prompt:

   ```
   cd %SOLR_HOME%\example
   java -Dbt.root=%BT_ROOT% -jar start.jar
   ```

   In a Unix shell, such as Bash:

   ```
   cd $SOLR_HOME/example
   java -Dbt.root=$BT_ROOT -jar start.jar
   ```

5. With your web browser, visit `http://localhost:8983/solr/admin/`

   a. Click the **ANALYSIS** link.

b. Next to the label **Field**, choose **type** from the first dropdown list.

c. Enter `text_ja` or `text_en` in the **Field name** field.

d. Enter text (Japanese or English, depending on which you have set up) in either or both of the two **Field value** fields.
For Japanese, you can copy and paste this sample text that includes a compound noun: 映画界では初の文化勲章を受賞. You can also copy and paste sentences from Japanese sites, such as `http://www.asahi.com` or `www.basistech.co.jp`.

e. Click the **Analyze** button.

You will see how Tokenizer and each Filter process the input text.

6. Refer to the **Solr** documentation and the web site for information about indexing actual documents.

6.2.1. Modifying the Solr Configuration

6.2.1.1. Removing Duplicate Tokens

The RLP Tokenizer may generate duplicate tokens when it is analyzing text. For example, if the text includes "fox", the token is "fox" and the stem token is also "fox". The inclusion of both adds to the size of the index but provides no benefit. If you want to avoid indexing duplicate tokens, include the following `<filter>` element in the `<analyzer type="index"` element of **schema.xml**:

```
<filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
```

Note: If the `postStem` attribute is not set to true, the analyzer does not generate a stem token for each word. In our configuration above, the `postStem` attribute does not appear in the query analyzer: the default setting is false.

6.2.1.2. Using `RLPTokenizerFactory` Attributes

As described in the Javadoc for RLPTokenizerFactory [api-reference/com/basistech/rlp/solr/RLPTokenizerFactory.html], there are a number of attributes that you can use to influence the behavior of the `RLPTokenizer`. You specify these attributes in the `<tokenizer>` element of the `<analyzer type="index"` and `<analyzer type="index"` elements in **schema.xml**. A few are described below:

lang. The `lang` attribute designates the language of the document to be indexed or the query: `en` for English, `ja` for Japanese, and the appropriate two-letter ISO639 language code for the other languages for which RLP supports base linguistic analysis. If you remove this attribute. RLP uses the RLI processor (which your RLP license must include) to identify the language. This works for indexing, but not very well for queries, since the text is too short to enable RLI to accurately determine the language.

postPartOfSpeech. By default, part-of-speech (POS) tags are stored with each token. The POS tags are used by `RLPPOSFilter` to filter the tokens that are returned. If you are not using `RLPPOSFilter`, you should turn off this attribute in the `RLPTokenizer` element: `postPartOfSpeech="false"`. If `postPartOfSpeech="false"`, be sure **schema.xml** does NOT include a `filter`element with `class="com.basistech.rlp.solr.RLPPOSFilterFactory"`; if this filter is present, RLP For Lucene will throw an Exception at runtime. *Note:* it is not a good idea to use POS tags in queries, since the query strings are too short to provide enough context for accurate part-of-speech tagging.

postWord. By default a token is generated for the surface form of each word (as possibly transformed by RCLU if the text is Japanese). If you only want a token for the dictionary form of the word, turn this

attribute off in the `RLPTokenizer` element: `postWord="false"`, and include `postLemma="true"` for Arabic text and `postStem="true"` for other languages.

6.2.1.3. Filtering POS Tags

`RLPPOSFilterFactory`

`RLPPOSFilterFactory` uses the `pos` attribute to designate a file name that lists the part-of-speech tags of the words to be indexed. **pos-en.txt** and **pos-ja.txt** in ***SOLR_HOME*/example/solr/conf** are the part-of-speech tag lists that Basis provides for English and Japanese. You can take a look at these files and comment out the part-of-speech tags that you want **Solr** to ignore, or remove "#" to activate part-of-speech tags currently commented out..

7. Working with Languages Other than English and Japanese

In addition to the English and Japanese analyzers, **RLP For Lucene** includes a generic analyzer: `RLPAnalyzer`. You can use `RLPAnalyzer` to analyze documents in any of the languages that RLP supports. See the Java reference for RLPAnalyzer [api-reference/com/basistech/rlp/lucene/RLPAnalyzer.html].

The analyzer requires an RLP context. Unless the context includes the Rosette Language Identifier (RLI), you must also identify the language. See "Creating an RLP Application: Defining an RLP Context" in the *RLP Application Developer's Guide*. For information about the properties you can set in an RLP context for a given language, see "RLP Processors" in the same manual.

You can also adjust the tokens that `RLPTokenizer` generates. See the Java reference for RLPTokenizer [api-reference/com/basistech/rlp/lucene/RLPTokenizer.html]. If you are using **SOLR**, you can adjust these settings in **schema.xml**. See Using `RLPTokenizerFactory` Attributes [9] .

If you are using the `RLPPOSFilter` [2] , to eliminate words of unwanted parts of speech, see the available part-of-speech (POS) tags for each language in "Appendix: Part-of-Speech Tags" in the *RLP Application Developer's Guide*. The filter users a text file (one POS tag per line) to specify the POS tags of interest.

8. Creating Your Own Analyzer

Given the precise requirements of your search application, you may determine that the analyzers and filter we provide do not meet your needs. For example, you may need a filter that pays attention to criteria beyond POS tags. Or perhaps you do not want to use the same analyzers for both indexing and querying. You may want to create your own analyzers for languages other than English and Japanese, rather than using `RLPAnalyzer` as distributed.

This SDK includes the source code and an Ant build script that you can use to regenerate **rlplucene.jar**. You can add your own classes or modify the classes that we provide. Please keep in mind that Basis does not support this software once you have introduced modifications.

For queries, you may want to skip the base linguistic analysis that **RLP** provides, and assume that users tend to enter words in their dictionary form. Linguistic analysis is difficult to perform, takes time, and may not be accurate for short query strings. For example, you may want to use the Solr WhitespaceTokenizer and LowerCaseFilter for European languages.

RLPTokenizer. `RLPTokenizer` is designed to be very flexible and modification should not be necessary for most applications. You can control its behavior with its constructor and its setter methods. See the Javadoc for RLPTokenizer [api-reference/com/basistech/rlp/lucene/RLPTokenizer.html].

9. How To Build

btrlplucene.jar is provided in the ***BT_ROOT*/rlp/lib/*BT_BUILD*** directory. Assuming you have modified the source code in the `com.basistech.rlp.lucene` or `com.basistech.rlp.solr` package, this section tells you how to regenerate **btrlplucene.jar**.

To build from the source code, do the following.

In the Windows Command Prompt:

```
// Replace RLPLUCENE with the path to the rlplucene directory.
cd RLPLUCENE
ant -Dbt.root=%BT_ROOT% -Dbt.arch=%BT_BUILD% build
```

In a Unix shell, such as bash:

```
// Replace RLPLUCENE with the path to the rlplucene directory.
cd RLPLUCENE
ant -Dbt.root=$BT_ROOT -Dbt.arch=$BT_BUILD build
```

The build places **btrlplucene.jar** and **btrlpextra.jar** [2] in the **build** directory. When you run the Demos, Ant places the jars in **build** in front of the jars in **lib** on the classpath, so the jar you created is used. To use your jar with Solr, you must copy **build/btrlplucene.jar** to ***SOLR_HOME*/example/solr/lib**.

Javadoc. If you have added or modified Javadoc code comments, you may want to regenerate the Javadoc. To do so, call ant as described above, replacing `build` with `-Dver=`*`version`* `javadoc`, where *`version`* is the version designator you wish to assign to this modified software. For example:

```
ant -Dbt.root=$BT_ROOT -Dbt.arch=ia32-w32-msvc80 -Dver="My Alpha 0.1" javadoc
```

Ant places the Javadoc output in **rlplucene/doc/api-reference**.

10. About the Logger

This module uses **Simple Logging Facade for Java** (SLF4J) for logging of this module and the RLP core. See http://www.slf4j.org/.

Version 1.5.2 of SLF4J is bundled with this module.

SFL4J is not a logging system by itself but is a facade for various logging APIs. Using SFL4J, the developer or an administrator can determine which one of many popular logging systems to use at runtime.

This is done by including one and only one adapter jar on the classpath, such as **sfl4j-jdk14-1.5.2.jar**, for the logging system of your choice, and the jar of that logging system (if not `java.util.logging`, which is part of the JDK). You also need to include the SFL4J API jar, **slf4j-api-1.5.2.jar** on the classpath.

The **rlplucene/build.xml** Ant script sets the classpath to use **sfl4j-jdk14-1.5.2.jar**. To use a different adapter, use `-Dslf4jimpl_jar=`*`adapter.jar`* when you call Ant, where *`adapter.jar`* is the path to the adapter jar.

[2] You should not need to modify the class file that appears in **btrlpextra.jar**.

The provided **build.xml** file also uses **conf/jul.properties** to configure `java.util.logging`. You may edit this file to change the logging level. See http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/package-summary.html and the class descriptions in this package for details.

Solr is hard-coded to use the logger in `java.util.logging`. To configure the log level for **Solr**, add this argument to the `java` command line before `-jar`: `-Djava.util.logging.config.file=slr/conf/jul.properties`

11. Directory Structure

When expanded to the same directory (preferably *`BT_ROOT`*), the **RLP Lucene** SDK and documentation packages display the following tree structure:

rlplucene
The root directory of **RLP For Lucene**.

build
If you rebuild from the source code, contains jars and classes.

conf
Configuration files.

doc
The **RLP For Lucene** documentation.

lib
Contains prebuilt RLP For Lucene and third-party jar.

queries
UTF-8 sample query files.

sampledocs
Contains language-specific directories with UTF-8 documents for indexing.

src
Source files for **RLP For Lucene**

com
The integration code, consisting of two packages, `com.basistech.rlp.lucene` and `com.basistech.rlp.solr`.

lia
Contains a utility class that accompanies the book *Lucene In Action*, Otis Gospodnetic & Erik Hatcher, Manning, ISBN 1-932394-28-1. This is licensed under Apache License, version 2.0. <http://www.apache.org/licenses/LICENSE-2.0.html>

org
A partial copy of demo program source code from the **Lucene** 2.0 distribution. These files are slighlty modified to use the RLP analyzer and to read files in UTF-8:
- **org/apache/lucene/demo/FileDocument.java**
- **org/apache/lucene/demo/IndexFiles.java**
- **org/apache/lucene/demo/SearchFiles.java**

These files are licensed under Apache License, version 2.0. <http://www.apache.org/licenses/LICENSE-2.0.html>.