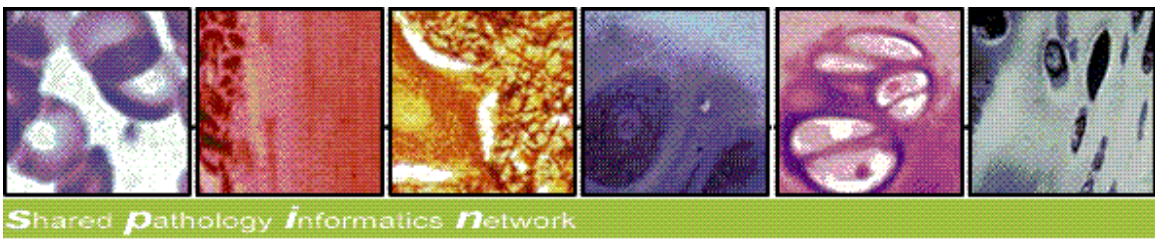
User Guide

SPINScrubberUserGuide

Title:

Author: Andrew McMurry

Contact: Andrew_McMurry(@)hms.harvard.edu



Intended Audience :

Technical staff of all levels should be able to configure this module for use with their records.
Programming experience is NOT required, though a basic understanding of XML is needed to edit the configuration file.

Scrubber Overview :

This scrubber utility removes confidential identifiers from structured XML or plain text by comparing the input text phrases to a list of known identifiers (names, states, etc) and a series of Regular Expressions.

While typically used to prepare confidential reports to be compliant with HIPAA standards, this utility is practical for any organization looking to protect privacy of their records – regardless if they are being used for medical purposes or not.

Required Software:

Java 1.5

Running the Scrubber Program :

Usage:
scrubber *InputFile(s) [ConfigurationFile]*

Where

- The *InputFile(s)* argument is either a file or directory
- The *ConfigurationFile* argument is optional.
  By default, scrubber will search for ScrubberConfiguration.xml in the same directory as the jar program.

Example 1:
scrubber c:\private\rawfiles\2006 conf\DefaultTextScrubber.xml

Example 2:
scrubber c:\private\xmlfiles\2006 conf\DefaultXMLTextScrubber.xml

Configuring the Scrubber Program :

The scrubber processes *batches* of either plain text files or XML files.
The instructions for processing each type are defined in a scrubber configuration file.

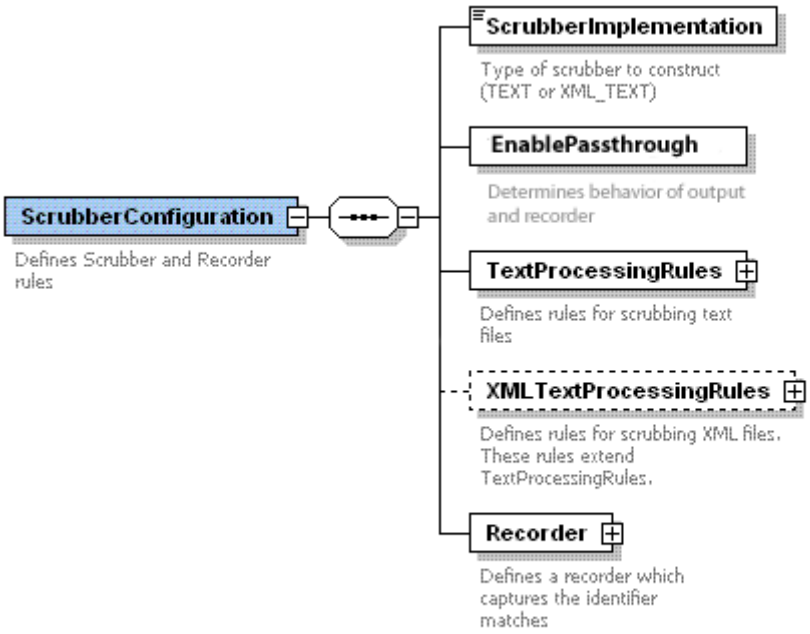
Provided Sample Configurations
For your convenience, sample configurations have been provided for you. Most users will simply apply one of the samples without any modifications needed. Both samples provide a tested list of identifiers and regular expressions.

Defa
ultT
extS
crub
ber.
xml

For plain text files, use:

D
ef
a
ul
t
X
MLTextScrubber.xml
(
S
cr
u
b
s
al
l
X
For XML files, use: ML text & CDATA nodes)

Customizing Scrubber Behavior: Advanced Configuration Settings



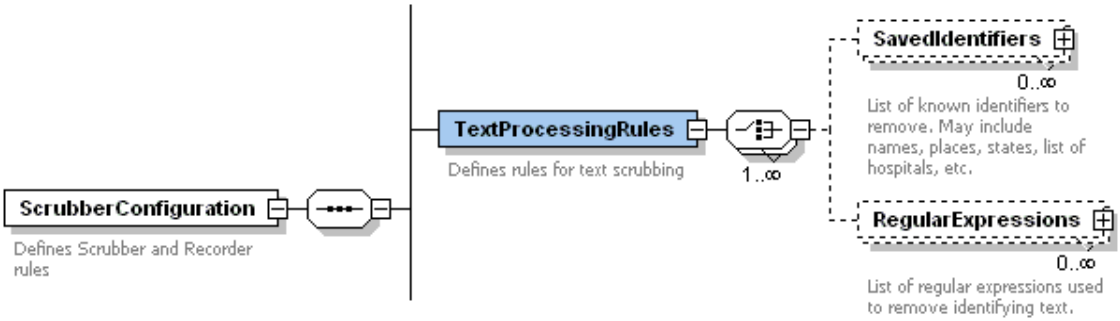
- The **Scrubber Implementation** denotes if the scrubber should expect XML files or Plain text Files. Valid settings are DEFAULT, XML_TEXT, or TEXT.
- The **EnablePassthrough** is a Boolean field. A value of **true** will disable scrubbing and instead send additional information to the Recorder identifying the character position and type of match. A value of **false** will allow scrubbing as normally expected.
- The **TextProcessingRules** defines the rules for text scrubbing.

- The **XMLTextProcessingRules** defines rules for scrubbing XML files.
  These rules extend TextProcessingRules.

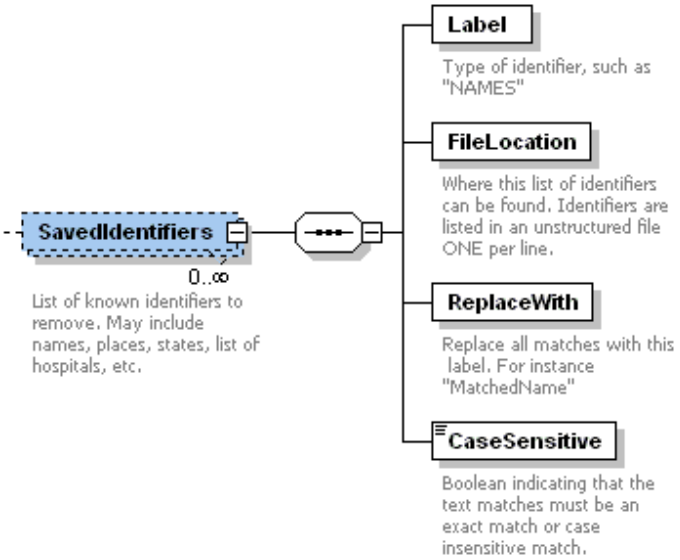
- The **Recorder** captures the identifier matches

The **TextProcessingRules** defines the basic rules for scrubbing both unstructured text and XML files, which are defined by testing a list of *SavedIdentifiers* and *RegularExpressions* .

At least one set of identifiers or regular expressions must be provided.



SavedIdentifiers contain a list of known identifiers which must be removed.
This can be places, states, Titles such as PHD or any other word you would like removed from the input.



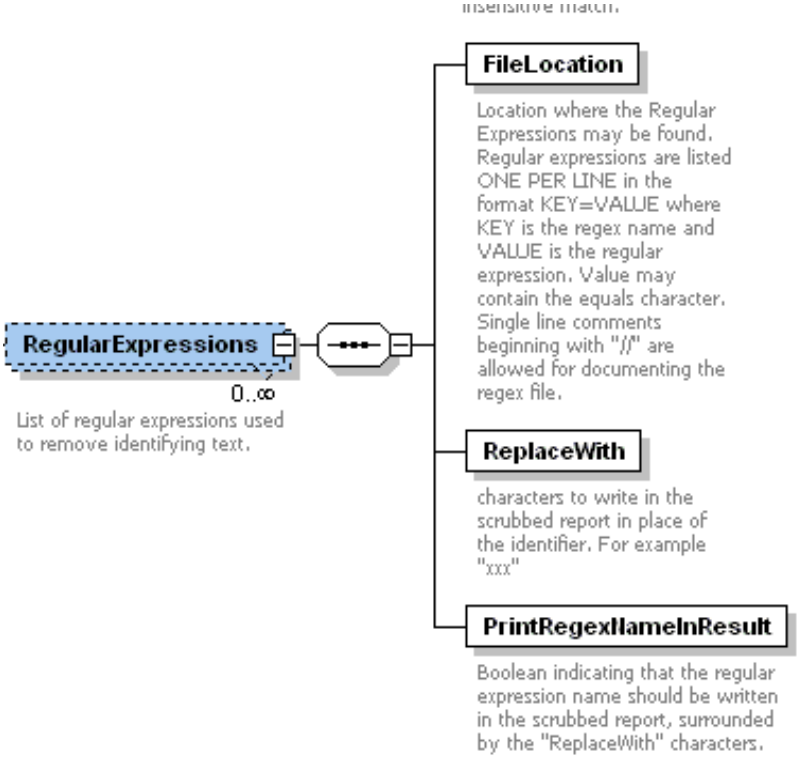
Label
Type of identifier, such as "states"

FileLocation
Where this list of identifiers can be found. Identifiers are listed in an unstructured file ONE per line.

ReplaceWith
Replace all matches with this text. For instance "StateName"

CaseSensitive
Boolean indicating that the text matches must be an exact match or case insensitive match.

The **RegularExpressions** are tested one by one against the text input.



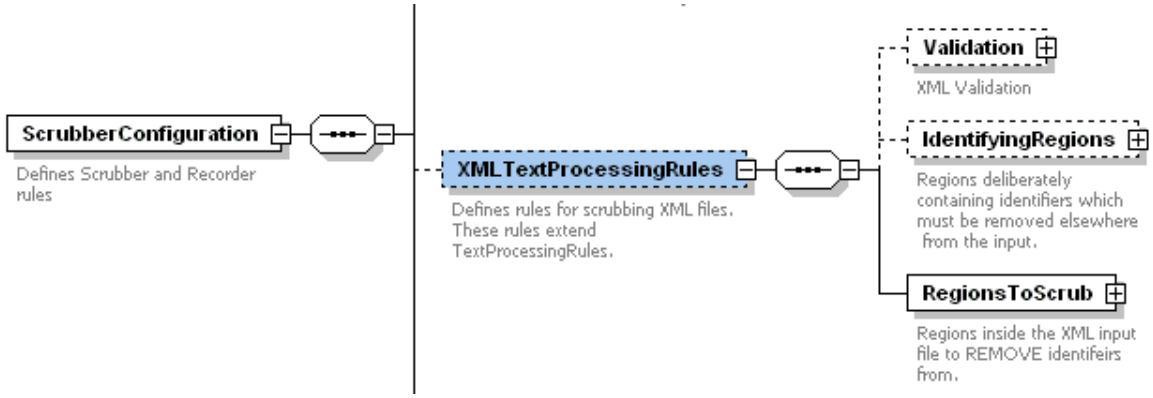
FileLocation
Location where the Regular Expressions may be found. Regular expressions are listed ONE PER LINE in the format KEY=VALUE where KEY is the regex name and VALUE is the regular expression. The VALUE may contain the equals “=” character. Single line comments beginning with "//" are allowed for documenting the Regex file.

ReplaceWith
Characters to write in the scrubbed report in place of the identifier. For example "xxx".

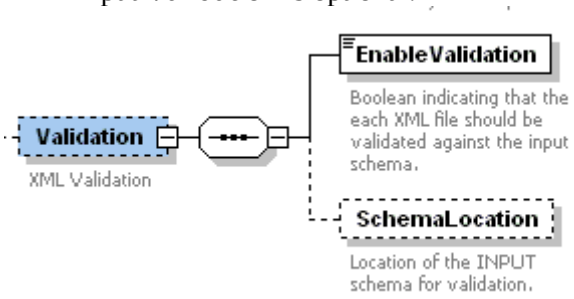
PrintRegexNameInResult
Boolean indicating that the regular expression name should be written in the scrubbed report, surrounded by the "ReplaceWith" characters.

The regular expression configuration file regex.list should be fined tuned to your particular medical record structure. An extensive library of regular expressions exist at http://regexlib.com/ and a Java-based online regular expression testing tool is located at: http://myregexp.com/

XMLTextProcessingRules defines rules for scrubbing XML files. These rules extend TextProcessingRules. The sample configuration DefaultXMLTextScrubber.xml disables validation, uses no IdentifyingRegions, and scrubs every element text/cdata node.



XML Input **Validation** is optional.

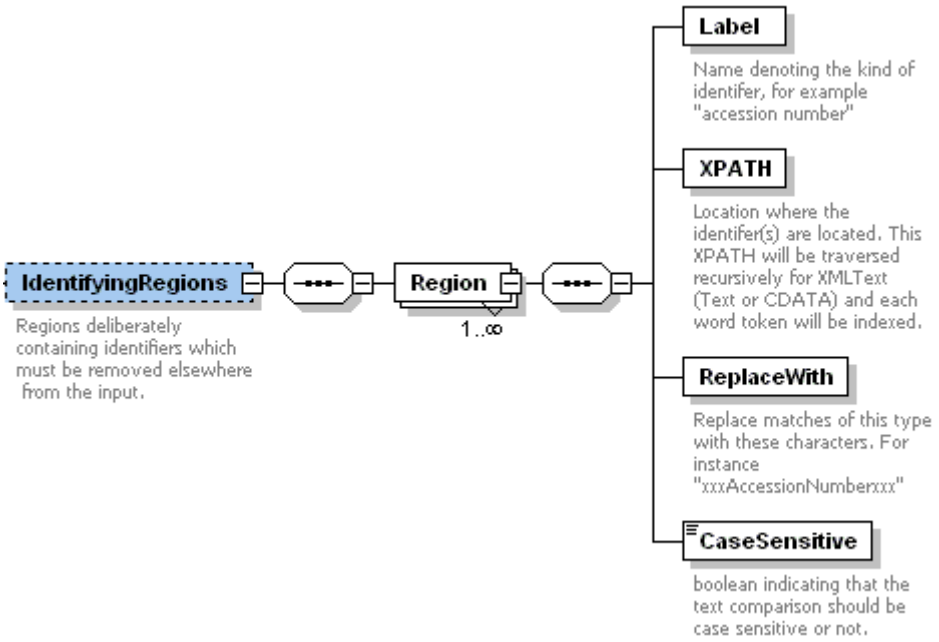


EnableValidation
Boolean indicating that the each XML file should be validated against the input schema.

SchemaLocation
Location of the INPUT schema for validation.
Note, this is the schema of the INPUT files, not the configuration file.
Again, if you want to use input validation it must match YOUR input schema.

IdentifyingRegions contain identifiers which are KNOWN to potentially exist elsewhere in the input file. This

enables the scrubber to search for *input specific identifiers* – eg, identifiers which are specific ONLY to a single record. This may include the patient's name, accession number, medical record number, etc.



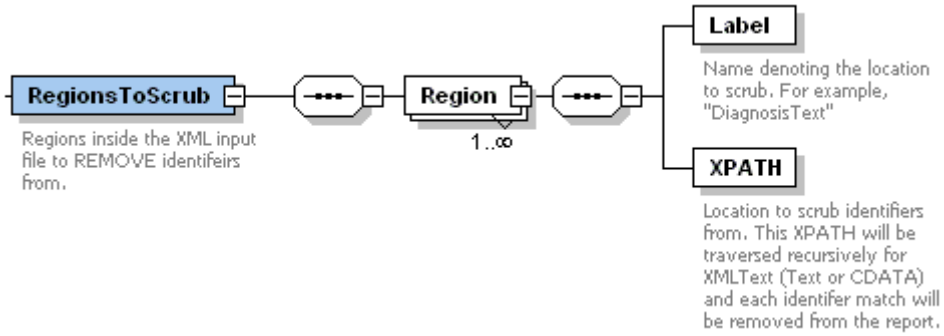
Label
Name denoting the kind of identifier, for example "accession number"

XPATH
Location where the identifer(s) are located. This XPATH will be traversed recursively for XMLText (Text or CDATA) and each word token will be indexed.

ReplaceWith
Replace matches of this type with these characters. For instance "xxxAccessionNumberxxx"

CaseSensitive
boolean indicating that the text comparison should be case sensitive or not.

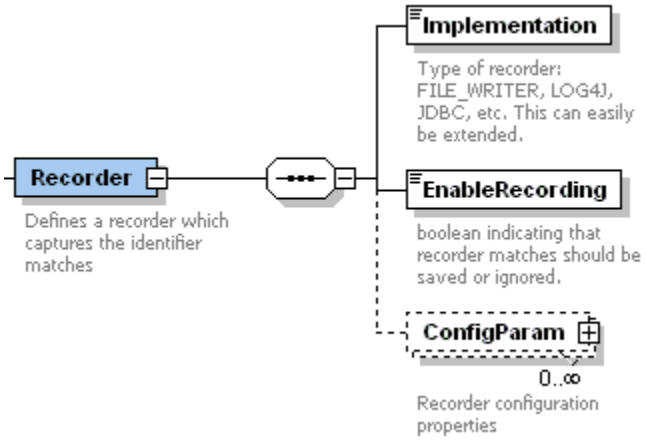
RegionsToScrub optimizes the scrubber by only targeting the XML nodes which you would like to remove identifiers from. This also allows nodes to be spared from scrubbing if those regions should pass through unmodified. The default sample configuration file will scrub ALL element text and CDATA by using an XPATH which starts at the document root.



Label
Name denoting the location to scrub.
For example, "Diagnosis".

XPATH
Location to scrub identifiers from.
This XPATH will be traversed recursively for XML Text or CDATA and each identifer match will be removed from the report.

A **Recorder** will save the text strings which were flagged as containing identifiers.
This is useful for both debugging and emerging at a near-perfect dictionary of identifying words.



Implementation
Type of recorder: FILE_WRITER, LOG4J, JDBC, etc. This can easily be extended.

EnableRecording
boolean indicating that recorder matches should be saved or ignored.

ConfigParam
Recorder configuration properties. For example, database connection related info if using the JDBC recorder.

Dictionaries (names.txt and legacyscrubber.list) The default dictionary for the Scrubber is names.list. It contains first and last names from the 2000 US Census minus common English words that could ambiguously be also last and first names. You can increase the size of the dictionary if you find the Scrubber missing some words. The legacy.list dictionary is included which contains an extended list of names from the previous Scrubber release but you may find that using over-scrubs too much data from the records.