



## CSV Transformer Documentation

Key	Value
Author:	Reto Scheiwiller
Last modified:	06.11.2015
CSV Comparator Version:	0.5

## Table of Contents

1	Introduction .....	3
1.1	History .....	3
2	CSVTransformerCLI.jar – Command Line Interface.....	4
2.1	Arguments .....	4
2.2	Examples.....	6
2.2.1	Example – Mandatory Input .....	6
2.2.2	Example – Transform Folder.....	7
2.2.3	Example – Using a Configuration File.....	8
3	Tips & Tricks .....	9
3.1	Unix: Copy .xml-Files .....	9
3.2	Unix: Copy configuration from .jar-Files.....	9
3.3	Automation with Bash-Script.....	10

# 1 Introduction

The CSV Transformer is a data processing tool which transforms .xml-Files to comma separated values. The format of the .xml-Files has to be valid, but schema definitions of the files will be ignored.

The tool takes either a single .xml-File file or folder containing .xml-Files, if a folder is specified all the .xml-Files in the folder will be transformed and merged to a single file.

The list of arguments for the tool can be provided by a config file using the argument “-config.file”.

## 1.1 History

The CSV Transformer was created in a load and performance testing project, the use case was to be able to transform 2800 configuration files of a big banking application to a single .csv-File. With this single file it was possible to compare the whole configuration between two releases with the already available tool CSV Comparator. This enabled the team to verify if there were any changes that could have a potential performance impact.

## 2 CSVTransformerCLI.jar – Command Line Interface

(required java version: 6.0)

### 2.1 Arguments

If no arguments or wrong arguments are provided through the command line, a list of available arguments will be printed. The only mandatory argument is “-input.path={filepath}”, all other arguments have default values.

Argument	Description
<b>-config.file</b>	<b>Syntax:</b> -config.file={filepath} The path to a config-file. The config-file can include all the arguments defined in this list delimited by newline. Also lines starting with “#” are considered as comments, as well blank lines are allowed.
<b>-config.loglevel.console</b>	<b>Syntax:</b> config.loglevel.console={ALL TRACE DEBUG INFO WARN ERROR FATAL OFF} <b>Default:</b> “INFO” Log level(log4j2) printed to the standard output
<b>-config.loglevel.file</b>	<b>Syntax:</b> config.loglevel.file={ALL TRACE DEBUG INFO WARN ERROR FATAL OFF} <b>Default:</b> “DEBUG” Log level(log4j2) printed to the logfile.
<b>-input.path</b>	<b>Syntax:</b> -input.path={filepath} The path to the input file or folder which should be transformed. If a folder is used, all files which match the regex set by -input.filefilter will be transformed.
<b>-input.format</b>	<b>Syntax:</b> -input.format={xml} <b>Default:</b> xml The format the input currently has and should be transformed into comma separated values. (currently only xml is available)
<b>-input.filefilter</b>	<b>Syntax:</b> -input.filefilter={regextoMatch} <b>Default:</b> “.*” Only files matching the specified regular expression will be transformed. The filter will be used only if a folder is specified with -input.path.
<b>-result.file</b>	<b>Syntax:</b> -result.file={filepath} <b>Default:</b> ./transform_result.csv”

	The path to the result file, where the compared data should be written.
<b>-result.delimiter</b>	<p>Syntax: -result.delimiter={delimiter}</p> <p>Default: “,”</p> <p>The delimiter which is used in the result to delimit the data.</p>
<b>-result.sort</b>	<p><b>Syntax:</b> -result.sort={true false}</p> <p><b>Default:</b> true</p> <p>If true, the result will be sorted by the natural order of the identifierColumn. If false, the rows from both files are merged by alternately taking one row of each file. (see argument “-column.identifier” for more information)</p>
<b>-result.stdout</b>	<p><b>Syntax:</b> -result.stdout={true false onerror}</p> <p><b>Default:</b> false</p> <p>If true, the result will be written to the standard output instead to a file. 'onerror' will print the result to standard output in case an error occurred during writing the file. If you plan to pipe the output to another tool, set '-config.loglevel.console=OFF' so nothing else will be written to the standard output.</p>
<b>-result.prefix</b>	<p><b>Syntax:</b> -result.prefix={prefix}</p> <p>This value will be used as a prefix in the column 'Key' in the result file for all results.</p>

## 2.2 Examples

### 2.2.1 Example – Mandatory Input

Only the argument “-input.path” is mandatory, for the other arguments there are default values or they are optional.

**Execute:** The execution from the command line will look like this:

```
java -jar CSVTransformer.jar -input.path="./testdata/smalltest.xml"
```

**Input:** Contents of smalltest.xml:

```
<root>
<test>
  <emptyelement />
  <attrOnly attribute="attributeOnly" />
  <attrAndContent attribute="bla" count="234" haha="hihi" >2015-03-09</attrAndContent>
  <subelements>
    <requestid>f30bb37c-5fdd-44d2-aec8-3d883f4a9ad4</requestid>
    <class>com.csvcomparator.cli.main.CSVComparatorCLI</class>
    <method>main</method>
  </subelements>
  <msg>
    multiline
    multiline
    multiline
    multiline
    multiline
  </msg>
  <cdatatest>
    <![CDATA[so many things]]>
    <![CDATA[]]>
  </cdatatest>
</test>
</root>
```

**Result:** Contents of “./transform\_result.csv”:

```
Key,Value
smalltest.xml.root.test.attrAndContent,attribute=bla / count=234 / haha=hihi / text=2015-03-09
smalltest.xml.root.test.attrOnly,attribute=attributeOnly
smalltest.xml.root.test.cdatatest.cdata,so many things
smalltest.xml.root.test.cdatatest.cdata,
smalltest.xml.root.test.msg,  multiline  multiline  multiline  multiline  multiline
smalltest.xml.root.test.subelements.class,com.csvcomparator.cli.main.CSVComparatorCLI
smalltest.xml.root.test.subelements.method,main
smalltest.xml.root.test.subelements.requestid,f30bb37c-5fdd-44d2-aec8-3d883f4a9ad4
```

#### Result Explanations:

- The result is written to the default result file “./transform\_result.csv”
- The default delimiter is comma “,”
- The column “Key” represents the structure of the file.
- The column “Value”:
  - Contains only the inner text of the element
  - Or attributes and the text of the element delimited by “/”
  - Newlines are always replaced by blanks
- The output is sorted by the column “Key”

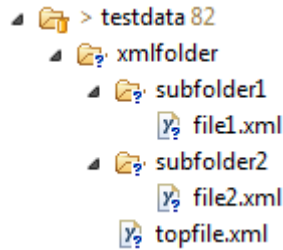
## 2.2.2 Example – Transform Folder

If the argument “-input.path” is a folder, all the files in the folder matching the regex defined by “-input.filefilter” (default is all >> “.\*”) will be transformed and stored in a single output file.

**Execute:** The execution from the command line will look like this:

```
java -jar CSVTransformer.jar -input.path=../testdata/xmlfolder
```

**Input:** The folder structure looks like this:



```
testdata
├── xmlfolder
│   ├── subfolder1
│   │   └── file1.xml
│   ├── subfolder2
│   │   └── file2.xml
│   └── topfile.xml
```

**Result:** Contents of “./transform\_result.csv”:

```
Key,Value
xmlfolder.subfolder1.file1.xml.root.description,folder test
xmlfolder.subfolder1.file1.xml.root.folder,id=sub1
xmlfolder.subfolder2.file2.xml.root.description,Test folder structure
xmlfolder.subfolder2.file2.xml.root.folder,id=sub2
xmlfolder.topfile.xml.root.description,In parent folder
xmlfolder.topfile.xml.root.folder,id=parent
```

### Result Explanations:

- The result is written to the default result file “./transform\_result.csv”
- The default delimiter is comma “,”
- The first part of the key represents the folder structure including the file name, the second part represents the node hierarchy of the file.
- The output is sorted by the column “Key”

### 2.2.3 Example – Using a Configuration File

**Execute:** The execution from the command line will look like this:

```
java -jar CSVComparator.jar -config.file="./testdata/transform.config"
```

#### Config-File:

Contents of “./testdata/transform.config”:

```
#####  
# Config File Test  
#####  
-input.path=./testdata/smalltest.xml  
-result.delimiter=;  
-result.sort=false  
-result.prefix=MyPrefix  
-result.file=./result/testConfigFile_result.csv
```

**Input:** Contents of smalltest.xml:

```
<root>  
<test>  
  <emptyelement />  
  <attrOnly attribute="attributeOnly" />  
  <attrAndContent attribute="bla" count="234" haha="hihi" >2015-03-09</attrAndContent>  
  <subelements>  
    <requestid>f30bb37c-5fdd-44d2-aec8-3d883f4a9ad4</requestid>  
    <class>com.csvcomparator.cli.main.CSVComparatorCLI</class>  
    <method>main</method>  
  </subelements>  
  <msg>  
    multiline  
    multiline  
    multiline  
    multiline  
    multiline  
  </msg>  
  <cdatatest>  
    <![CDATA[so many things]]>  
    <![CDATA[]]>  
  </cdatatest>  
</test>  
</root>
```

**Result:** Contents of “./result/testConfigFile\_result.csv”:

```
Key;Value  
MyPrefix.smalltest.xml.root.test.attrOnly;attribute=attributeOnly  
MyPrefix.smalltest.xml.root.test.attrAndContent;attribute=bla / count=234 / haha=hihi / text=2015-03-09 07:23:26.446  
MyPrefix.smalltest.xml.root.test.subelements.requestid;f30bb37c-5fdd-44d2-aec8-3d883f4a9ad4  
MyPrefix.smalltest.xml.root.test.subelements.class;com.csvcomparator.cli.main.CSVComparatorCLI  
MyPrefix.smalltest.xml.root.test.subelements.method;main  
MyPrefix.smalltest.xml.root.test.msg; multiline multiline multiline multiline multiline  
MyPrefix.smalltest.xml.root.test.cdatatest.cdata;so many things  
MyPrefix.smalltest.xml.root.test.cdatatest.cdata;
```

#### Result Explanations:

- The result is written with semicolon “;” as delimiter.
- The string “MyPrefix” is added to the keys
- The rows are not sorted and the order is as in the xml file



## 3 Tips & Tricks

### 3.1 Unix: Copy .xml-Files

To copy all .xml-Files from a directory and it's subdirectories, you can use rsync like this:

```
rsync -r --include="*/" --include=".xml" --exclude="*" <source_dir> <target_dir>
```

### 3.2 Unix: Copy configuration from .jar-Files

To copy the configuration from .jar-Files you can extract the files with the jar-tool and copy them to a target location:

```
TMPDIR="./tmp"
TARGETDIR="./target"
JARDIR="/app/bin/jars"

for JAR in `find ${JARDIR} -name *jar`; do

    JARNAME=`basename ${JAR}`

    #-----
    # Extract .ear-File
    mkdir ${TMPDIR}/${JARNAME}
    cd ${TMPDIR}/${JARNAME}

    /java/jdk7/bin/jar xf ${JAR} "META-INF/weblogic-application.xml" "META-
INF/application.xml"

    #-----
    # Copy to Target dir

    cp ${TMPDIR}/${JARNAME}/META-INF/weblogic-application.xml
    ${TARGETDIR}/${JARNAME}.weblogic-application.xml
    cp ${TMPDIR}/${JARNAME}/META-INF/application.xml
    ${TARGETDIR}/${JARNAME}.application.xml

done
```

### 3.3 Automation with Bash-Script

You can automate the call to the jar-file with a bash script like this:

```
#!/bin/bash
#####
# Author: Reto Scheiwiller
#
# Compare Files
# -----
# This script will transform the .xml-Files defined by the first argument of the script.
#
#####

# Check if there are no arguments present
if [ "$#" -ne 1 ]
then
#####
# Print Help
#####
echo "compare_CSV.sh takes no arguments, set the variables in the script."

exit 1
else

#####
# Set Configuration
#####

#### Java Configuration ####
JAVA=/cygdrive/c/Program Files (x86)/Java/jre1.6.0/bin/java
CSVTRANSFORM_JAR=C:\Programs\CSVTransformer\CSVTransformer.jar

INPUT_PATH=$1

#Result Config
RESULT_FILE=".\\transform_result.csv"
RESULT_DELIMITER=";"

#####
# Compare Files
#####

"$JAVA" -jar $CSVTRANSFORM_JAR \
    "-input.path=${INPUT_PATH}" \
    "-result.file=${RESULT_FILE}" \
    "-result.delimiter=${RESULT_DELIMITER}"

fi
```