Java Shapefile Tool

Configuration and User Guide

October 2017

Version 1.0

Table of Contents

1. Introduction 2

2. Installation 3

2.1 System Requirements 3

2.1.1 Command Prompt Use Requirements 3

2.1.2 Database Use Requirements 3

2.2 Steps to Install 3

2.2.1 Command Prompt Use Installation 3

2.2.2 Database Use Installation 4

2.3 Installing Java Shapefile Tool Database Components 5

3. Usage 6

3.1 Parameters 6

3.1.1 Table 1 – Java Shapefile Tool Parameters 6

3.1.2 Table 2 – Java Shapefile Extractor Parameters 7

3.1.3 Table 3 – Java Shapefile Uploader Parameters 9

3.2 Command-line Usage 10

3.3 Database Usage 11

4. Logging 13

4.1 Shapefile Log File 13

4.2 System Log File 13

4.3 Command/Database Level Log 13

4.3.1 Command Level Log 13

4.3.2 Database Level Log 13

5. GeoTools Source Code Changes 14

5.1 Class – `org.geotools.data.shapefile.ShapefileDataStore` 14

5.2 Class – `org.geotools.data.shapefile.dbf.DbaseFileWriter` 14

5.3 Class – `org.geotools.data.shapefile.shp.ShapefileReader` 14

1. Introduction

Java Shapefile Tool is developed to allow extract and upload of Shapefiles from and to (respectively) Oracle database. This tool is a Jar file developed using APIs from GeoTools (13-beta) and Oracle Spatial Java API (11.2.0.2).

The tool can be used either as a command-line tool or by loading in Oracle database.

2. Installation

Staging folder is the folder containing this document.

2.1 System Requirements

2.1.1 Command Prompt Use Requirements

a. **Java JRE** – **JRE 1.7** or later must be installed on the system from which the tool is to be used.
b. **Database Server Access** – The system running the tool must have access to the database server on which extract and upload operations are to be performed.

2.1.2 Database Use Requirements

Oracle database in which the tool is to be loaded must have default Java version 1.7 or later (practically, Oracle Database 12c or later).

Use following command to get the default Java version of a database –

```
SELECT dbms_java.get_ojvm_property(PROPSTRING => 'java.version') java_version
   FROM DUAL;
```

2.2 Steps to Install

2.2.1 Command Prompt Use Installation

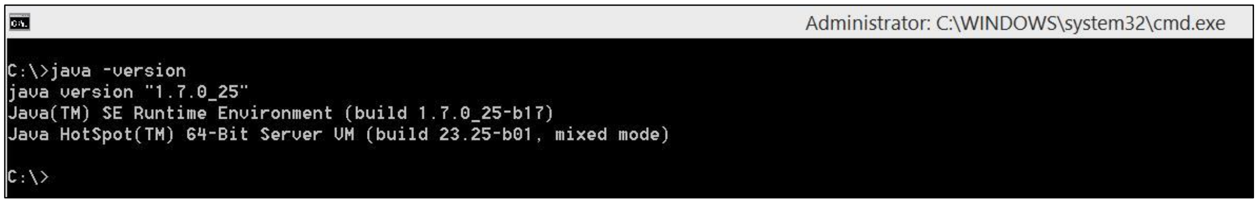
a. **Installing Java Run Time Environment (JRE)**

Install Java **JRE 1.7** on the system. While installing make sure that the installation directory **does not contain any white spaces**.

e.g. `C:\Java\64\jre1.7.0_25\`

Open a **Command Window** and run the following command –
`java -version`

It should give an output like shown in the following image. Environment variable `PATH` might need to be updated to set the correct Java path.



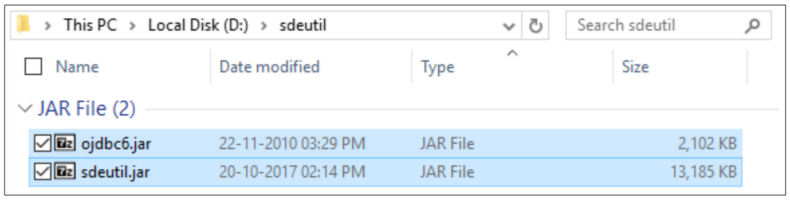
Administrator: C:\WINDOWS\system32\cmd.exe

```
C:\>java -version
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b17)
Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)

C:\>
```

b. **Copying the Jar files**

Create a directory on the system (e.g. `sdeutil`) at a suitable location (again not containing any white spaces). Copy `sdeutil.jar` and `ojdbc6.jar` from the staging folder to this directory.



Name	Date modified	Type	Size
JAR File (2)			
ojdbc6.jar	22-11-2010 03:29 PM	JAR File	2,102 KB
sdeutil.jar	20-10-2017 02:14 PM	JAR File	13,185 KB

c. **Setting shapefile-base-directory**

Create an Environment variable on client machine named – `SDE_UTIL_PATH` – and assign a path not containing any white spaces to it.

e.g. `D:\sdeutil\`

The path must already exist on the client machine.

This path will serve as base directory for the tool. If not set, location of `sdeutil.jar` will be used as base directory.

2.2.2 Database Use Installation

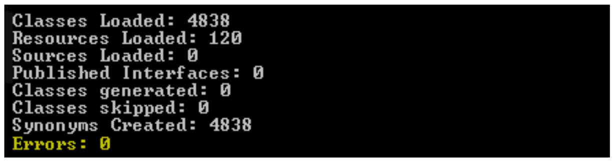
a. **Loading the Shapefile Java Tool in Oracle database**

Open a Command Window with staging folder as working directory and run following command –

```
loadjava_sdeutil.bat db_user/db_password@tns_name db_role
```

where `db_user`, `db_password`, `tns_name` are details to connect to database and `db_role` is role that need to be granted the privilege to execute the tool from database (you may use `PUBLIC` to grant the execute privilege to all the database users).

This command may take some time to load the tool in database. At the end of execution, make sure that the Error count is zero as shown in below image –



```
Classes Loaded: 4838
Resources Loaded: 120
Sources Loaded: 0
Published Interfaces: 0
Classes generated: 0
Classes skipped: 0
Synonyms Created: 4838
Errors: 0
```

b. **Setting shapefile-base-directory**

Create an Environment variable on database server named – `SDE_UTIL_PATH` – and assign a path not containing any white spaces to it.

e.g. `D:\sdeutil\`

The path must already exist on the database server.

This path will serve as base directory for the tool. If not set, `<ORACLE_HOME>` will be used as base directory.

2.3 Installing Java Shapefile Tool Database Components

This step is required for both command-line and database use. **Logon** to **SQL*Plus** with staging folder as working directory. At the prompt type –

```
START install_sdeutil.sql
```

In case public synonyms are required for other database users to use Java Shapefile Tool, run –

```
START create_sde_synonyms.sql
```

Exit SQL*Plus

Note 1 – Database user logging in must have following database privileges to complete the installation successfully –

```
CREATE TYPE
CREATE ROLE
CREATE TABLE
CREATE ANY INDE
CREATE SEQUENCE
CREATE TRIGGER
CREATE PROCEDURE
CREATE PUBLIC SYNONYM
```

3. Usage

3.1 Parameters

Following tables list all the parameters and their descriptions for the Java Shapefile Tool.

3.1.1 **Table 1** – Java Shapefile Tool Parameters

Parameter	Description
-help	To see the command line usage of Java Shapefile Tool
-setup	To setup shapefile-base-directory [Note 2]
-sde2shp	To extract a shapefile, followed by shapefile extractor parameters [Table 2]
-shp2sde	To extract a shapefile, followed by shapefile uploader parameters [Table 3]

Note 2 – Running Java Shapefile Tool with `–setup` option will create following directory structure in shapefile-base-directory –

```
<shapefile-base-directory>
      |_ column_map
      |_ epsg_db
      |       |_ v8.6.0.0
      |       |_ EPSG.data
      |       |_ EPSG.properties
      |       |_ EPSG.script
      |       |_ EPSG_creation_marker.txt
      |_ extract
      |_ log
      |_ upload
```

Directory	Description
colum_map	Directory to keep column-name(attribute) mapping file (for extract and upload)
epsg_db	Directory for EPSG database (for tool use only)
extract	Directory where Shapefiles will be extracted (for extract only)
log	Directory for System Log File [Section 4.2] (for extract and upload)
upload	Directory where Shapefiles will be uploaded (for upload only)

3.1.2 **Table 2** – Java Shapefile Extractor Parameters

Parameter	Description
`-help`	Specify this option to see the command line usage of Shapefile Extractor
Mandatory Alternate Parameters Group 1	
	Either
`-nc`	Specify this option, if the jar is loaded in database and called from a PL/SQL procedure or function (no values for this parameter)
	Or
`-h`	Host machine name/IP with existing Oracle database
`-p`	Host machine's port with existing Oracle database (e.g. 1521)
`-s`	Host machine's SID with existing Oracle database
`-u`	Database user's username
`-d`	Database user's password
Mandatory Alternate Parameters Group 2	
	Either
`-t`	Input feature table name and spatial column name (separated by **comma** only)
`-f`	File name of an output Shapefile **without extension**
	Or
`-whelp`	WHERE Clause ID for which details are required (use ID as '**all**' for details of all available WHERE clauses)
Optional Parameters	
`-w`	WHERE Clause ID for the query followed by any parameters enclosed in **square brackets []** separated by **space.** In case parameter value contains space itself, enclose the value in **double quotes**. **None** of the values in [] can have square brackets in turn. [Note 3]
`-a`	File name containing column-name(attribute) mappings **with extension**

Note 3 – To avoid SQL-Injection and at the same time maintain the flexibility to use WHERE clause while extracting a Shapefile, a new way has been introduced. A table named – `SDE_WHERE` – gets created while installing the database components of the tool with following structure –

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------
SW_ID                                     NOT NULL NUMBER
SW_UNIQUE                                 NOT NULL VARCHAR2(50)
SW_WHERE_CLAUSE                           NOT NULL CLOB
SW_WHERE_DESCR                                     VARCHAR2(4000)
SW_USER_ROLE                              NOT NULL VARCHAR2(4000)
```

Where,

1. **`SW_ID`** is an index column, autogenerated through a sequence
2. **`SW_UNIQUE`** holds a 50 character long unique identifier, which will be used as a parameter for `–w` parameter while extracting a Shapefile
3. **`SW_WHERE_CLAUSE`** contains the actual WHERE clause definition. In case a user input is required, following placeholders can be used –

   a. `<<?NUMBER?>>` for number type input
   b. `<<?VARCAHR?>>` for character type input
   c. `<<?DATE?>>` for date type input

   e.g.
   ```
   TREE_AGE <= <<?NUMBER?>> AND TREE_TYPE = <<?VARCHAR?>> AND TREE_START_DATE
   <= <<?DATE?>>
   ```

4. **`SW_WHERE_SDECR`** is optional, contains maximum of 4000 characters long description for the corresponding WHERE clause
5. **`SW_USER_ROLE`** contains a database role name. If kept **null**, will get automatically populated with `SDE_USER` role. This indicates that user must have this role to use the WHERE clause to extract a Shapefile.

Database user with `SDE_ADMIN` role only can `insert`, `update`, `delete` records in this table. This is to ensure the security from SQL-Injection.

For now, there is no UI to update this table, so `SDE_ADMIN` must do it manually by firing `insert`, `update`, `delete` statements.

3.1.3 **Table 3** – Java Shapefile Uploader Parameters

Parameter	Description
`-help`	Specify this option to see the command line usage of Shapefile Uploader
Mandatory Alternate Parameters	
Either	
`-nc`	Specify this option, if the jar is loaded in database and called from a PL/SQL procedure or function (no values for this parameter)
Or	
`-h`	Host machine name/IP with existing Oracle database
`-p`	Host machine's port with existing Oracle database (e.g. 1521)
`-s`	Host machine's SID with existing Oracle database
`-u`	Database user's username
`-d`	Database user's password
Mandatory Parameters	
`-o`	Mode to add Shapefile data to a table. Possible Values - {**append**|**create**|**init**}
`-t`	Table name for the result
`-f`	File name of an input Shapefile **without extension**
Optional Parameters	
`-i`	Column name for unique numeric ID; if required
`-r`	Valid Oracle SRID for coordinate system; use 0 if unknown [Note 4]
`-g`	Preferred or valid **SDO_GEOMETRY** column name
`-x`	Bounds for the X dimension; use -180,180 if unknown
`-y`	Bounds for the Y dimension; use -90,90 if unknown
`-m`	Load tolerance fields (x and y) in metadata, if not specified, tolerance fields are 0.05
`-n`	Start ID for column specified in **`-i`** parameter
`-c`	Commit interval. Default, only commits at the end of a run.
`-a`	File name containing column-name(attribute) mappings **with extension**

Note 4 – In case of *no projection* (i.e. `prj` file) available for the Shapefile being uploaded, SRID must be specified using `-r` command line argument mentioned above. If not, upload process will be terminated.

The priority to get the SRID will be –
1. `prj` File
2. SRID specified with `-r` option

In both the above cases, if the mode to add Shapefile data is `append` or `init`, and SRID **differs** the present SRID for **table**, **geometry-column** combination (i.e. existing record in `user_sdo_geom_metadata` in database) of the system to which the Shapefile is being uploaded, upload process will be **terminated**.

3.2 Command-line Usage

The jar can be executed through a **Command Window** on system, like –

```
java -jar "D:\sdeutil\sdeutil.jar" [parameters]
```

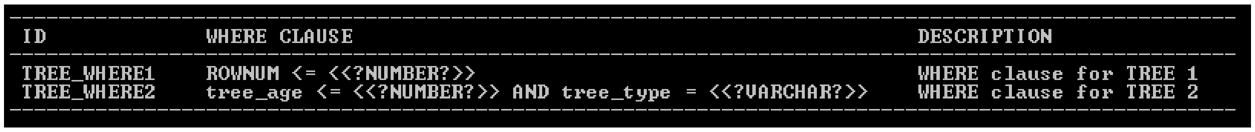
Here are few examples –

a. Command to setup shapefile-base-directory

```
java -jar "D:\sdeutil\sdeutil.jar" -setup
```

b. Command to see all available WHERE clauses

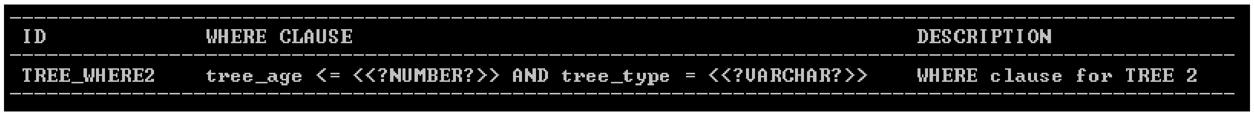
```
java -jar "D:\sdeutil\sdeutil.jar" -sde2shp -h db-server.example.com -p
1521 -s ORCL -u HIGHWAYS -d highways -whelp all
```



```
--------------------------------------------------------------------------------------------------------------------
ID             WHERE CLAUSE                                                DESCRIPTION
--------------------------------------------------------------------------------------------------------------------
TREE_WHERE1    ROWNUM <= <<?NUMBER?>>                                      WHERE clause for TREE 1
TREE_WHERE2    tree_age <= <<?NUMBER?>> AND tree_type = <<?VARCHAR?>>      WHERE clause for TREE 2
--------------------------------------------------------------------------------------------------------------------
```

c. Command to see a WHERE clause ID details

```
java -jar "D:\sdeutil\sdeutil.jar" -sde2shp -h db-server.example.com -p
1521 -s ORCL -u HIGHWAYS -d highways -whelp TREE_WHERE2
```



```
--------------------------------------------------------------------------------------------------------------------
ID             WHERE CLAUSE                                                DESCRIPTION
--------------------------------------------------------------------------------------------------------------------
TREE_WHERE2    tree_age <= <<?NUMBER?>> AND tree_type = <<?VARCHAR?>>      WHERE clause for TREE 2
--------------------------------------------------------------------------------------------------------------------
```

d. Command to extract a Shapefile

```
java -jar "D:\sdeutil\sdeutil.jar" -sde2shp -h db-server.example.com -p
1521 -s ORCL -u HIGHWAYS -d highways -t TREE_LOCATIONS,SHAPE -w TREE_WHERE2
[20 "MANGO"] -f TREE_LOCATIONS_EXTRACT -a extract_tree_loc_attributes.txt
```

e. Command to upload a Shapefile

```
java -jar "D:\sdeutil\sdeutil.jar" -shp2sde -h db-server.example.com -p
1521 -s ORCL -u HIGHWAYS -d highways -t TREE_LOCATIONS -f
TREE_LOCATIONS_UPLOAD -i TREE_ID -r 4326 -g SHAPE -x -180,180 -y -90,90 -m
0.05 -o create -n 1 -c 12 -a upload_tree_loc_attributes.txt
```

Note 5 – (`-h`, `-p`, `-s`, `-u`, `-d`) must be used from alternate mandatory parameters group 1 to inform the tool to create a database connection using the connection details passed.

3.3 Database Usage

Once the Java Shapefile Tool is loaded in the database, it can be called using a PL/SQL block. An Oracle **Collection TYPE** gets created while installing database components for the tool – `SDE_VARCHAR_ARRAY`.

This type must be used to configure the parameters required for the tool to execute a command. Below is an example to perform a Shapefile Extract –

```
DECLARE
     v_command sde_varchar_array := sde_varchar_array();
     v_result  VARCHAR2(32767);
BEGIN
-- SDE2SHP
     v_command.EXTEND(14);
     --
     v_command(1)  := '-sde2shp';
     v_command(2)  := '-nc';
     v_command(3)  := '-t';
     v_command(4)  := 'TREE_LOCATIONS,SHAPE';
     v_command(5)  := '-w';
     v_command(6)  := 'TREE_WHERE2';
     v_command(7)  := '[';
     v_command(8)  := '20';
     v_command(9)  := 'MANGO';
     v_command(10) := ']';
     v_command(11) := '-f';
     v_command(12) := 'TREE_LOCATIONS_EXTRACT';
     v_command(13) := '-a';
     v_command(14) := 'extract_tree_loc_attributes.txt';
     --
-- RESULT
     v_result := sde_util.shputil(v_command);
     dbms_output.put_line(v_result);
END;
/
```

Key Points –

1. `–nc` parameter must be used from the alternate mandatory parameters group 1 to inform the tool to use the same database connection as the one executing the PL/SQL block, called - nested database connection.

2. The command is similar to the one that gets formulated in command-line usage, just that each key and each value needs to be added to `SDE_VARCHAR_ARRAY` as a separate parameter. Check the formulation of `–w` parameter, two values (`20` and `MANGO`) are passed as two separate parameters, the square brackets can optionally be combined with start and end values, like –
```
v_command(7)  := '[20';
v_command(8)  := 'MANGO]';
```

3. The size of SDE_VARCHAR_ARRAY (i.e. `v_command.EXTEND(14)`) must match the number of parameters being passed to it.

4. The size of the variable holding the result of the command must be 32767 (e.g. `v_result VARCHAR2(32767)`).

5. The command needs to be called using a predefined function `sde_util.shputil` that gets created while installing the database components for the tool.

Here is an example to perform Shapefile Upload –

```
DECLARE
     v_command sde_varchar_array := sde_varchar_array();
     v_result  VARCHAR2(32767);
BEGIN
-- SHP2SDE
     v_command.EXTEND(26);
     --
     v_command(1)   := '-shp2sde';
     v_command(2)   := '-nc';
     v_command(3)   := '-t';
     v_command(4)   := 'TREE_LOCATIONS';
     v_command(5)   := '-f';
     v_command(6)   := 'TREE_LOCATIONS_UPLOAD';
     v_command(7)   := '-i';
     v_command(8)   := 'TREE_ID';
     v_command(9)   := '-r';
     v_command(10)  := '4326';
     v_command(11)  := '-g';
     v_command(12)  := 'SHAPE';
     v_command(13)  := '-x';
     v_command(14)  := '-180,180';
     v_command(15)  := '-y';
     v_command(16)  := '-90,90';
     v_command(17)  := '-m';
     v_command(18)  := '0.05';
     v_command(19)  := '-o';
     v_command(20)  := 'create';
     v_command(21)  := '-n';
     v_command(22)  := '1';
     v_command(23)  := '-c';
     v_command(24)  := '12';
     v_command(25)  := '-a';
     v_command(26)  := 'upload_tree_loc_attributes.txt';
     --
-- RESULT
     v_result := sde_util.shputil(v_command);
     dbms_output.put_line(v_result);
END;
/
```

4. Logging

There are three logging levels depending on when and where an error encounters –

4.1 Shapefile Log File

It refers to a log file in the folder where the Shapefile being extracted (i.e. `<shapefile-base-directory>\extract\`) or uploaded (i.e. `<shapefile-base-directory>\upload\`) and having same name as the Shapefile under operation. It contains information about various steps being followed while executing the extract or upload of the Shapefile and any errors encountered while executing the command.

This logging level generally logs errors related to wrong parameter-values and errors at database level like wrong username/password, table does not exist, shape column not found etc.

4.2 System Log File

It refers to a log file contained in folder – `<shapefile-base-directory>\log\`.
This log file contains a list of parameters and their values passed to the command (mentioned the tables of Section 4.2).

This logging level generally logs errors related to passing wrong number of parameters, absence of files mentioned in the command and those that are thrown before creation of Shapefile Log file.

4.3 Command/Database Level Log

4.3.1 Command Level Log

It refers to the command line output while executing the extract/upload process. In case of successful execution, the output shows –

i. Complete path to the System Log file.
ii. A message – “success” – indicating the process completed successfully. This message can be used in the *PL/SQL way* as explained in Section 4.2.

This logging level generally logs errors related to abnormal termination of the extract/upload process and those that are thrown before creation of System Log file.

4.3.2 Database Level Log

It’s same as command level log, just that any success/error messages are returned as a string (at max 32767 characters long) by `sde_util.shputil` function instead of printing on console.

5. GeoTools Source Code Changes

GeoTools being an open source API, below mentioned Java class files were edited to suit the requirements.

5.1 Class – `org.geotools.data.shapefile.ShapefileDataStore`

`ShapefileDataStore` – was forcing the data type `NUMBER` to be `NUMBER (33, 31)`. This was causing problem as it allows only two digits before decimal point and hence truncating numbers in essential records of database.

It is changed to make it – `NUMBER (19, 9)`.

Precision and scale were selected to be 19 and 9 respectively to match the ESRI Shapefile specifications.

5.2 Class – `org.geotools.data.shapefile.dbf.DbaseFileWriter`

`DbaseFileWriter` – was replacing a NULL Date with eight '0's – default length for a Date type column in a Shapefile being eight characters.

It is changed to show null dates as `NULL` only instead of '0's.

5.3 Class – `org.geotools.data.shapefile.shp.ShapefileReader`

`ShapefileReader -> private boolean hasNext(boolean checkRecno)` method was not checking for `null shxReader` (i.e. Shapefile Index file reader object), which was causing `java.lang.NullPointerException` at some places unwantedly.

The check for `null shxReader` is added.