# Scripting SandMark

SandMark can be run through a CLI interface, or from a shell prompt.

---

Use the following command to run CLI:
java -cp sandmark.jar sandmark.smash.SandmarkCLI -options

**CLI options:**
-U : Usage
-S : Display static watermark list
-D : Display dynamic watermark list
-R : Display obfuscator list
-L <level> : Specify obfuscator level to be displayed - 'app'/'class'/'method'
-A <algoName> : Specify algorithm
-H : Get algorithm help
-i <inputJar> : Specify input jar file
-o <outputJar> : Specify output jar file
-w <watermark> : Specify watermark value
-k <key> : Specify key value
-E : Perform embedding
-R : Perform recognizing
-O : Perform obfuscation
-P <prefixStr> : Specify prefix string(for ConstantString watermark)

**Dynamic Watermark specific options:**
-t : Specify trace file
-c : Set ClassPath
-m : Set MainClass
-a : Set Arguments
-T : start Trace
-N <encodeStr> : Specific parameters for CT/DynamicAA algorithm.

encodeStr format -

- CT:
  <nodetype>#<storewhat>#<storemethods>#<storelocation>#<protectionmethods>#<encoding>#<components>#<encodeInline(true/false)>#<encodeReplace(true/false)>#<dumpIR(true/false)>
- DynamicAA/ Arboit
  Algorithm: <minWMparts>#<encodeConsts(true/false)>#<useopaquemethods(true/false)>#<reusemethods(true/false)>

---

Use the following command to start up SandMark shell:
java *-cp sandmark.jar* sandmark.smash.Smash

At the ***$smash*** prompt, you can give the following shell commands:

- **help(); -**Get help for various options
- **swm(); -** Display static watermark algorithm list
- **dwm(); -** Display dynamic watermark algorithm list
- **obf(level); -** Display obfuscation algorithm  list at 'app', 'class' or 'method' level.
- **addOptions(); -** Display additional options;
- **setInputFile(jarfile); -** Set the input jar file name, specifying its fullpath
- **setOutputFile(jarfile); -** Set the output jar file name; if not set, the default is in_wm.jar for watermark and in_obf.jar for obfuscation, where in.jar is the input file.
- **setKey(key); -**Set the key value for the watermark algorithm
- **setWM(watermark); -** Set the watermark value
- **selectAlg(algoName); -** Select the algorithm
- **getAlgDesc(); -** Get description of the current algorithm selected; selectAlg() must be run before this.
- **getAlgDesc(algoName); -**Get description of algorithm specified
- **embed();  -** Perform embed operation
- **recognize(); -** Perform recognition operation
- **getNextWM(); -** Retrieve the next watermark after recognition
- **setTraceFile(tracefile); -** Specify the file of trace-points generated during tracing
- **setClassPath(classpath); -** Specify user defined ClassPath additions used during tracing or recognition
- **setMainClass(mainclass); -** Specify mainClass where execution starts during tracing or recognition
- **setArguments(args); -** Set arguments during tracing or recognition
- **startTrace(); -** start tracing
- **stopTrace(); -** stop tracing
- **endTrace(); -** terminate tracing
- **ct_encode(nodetype, storeWhat, storeMethods, storeLoc, protectionMethods, encoding, components); -** specify encoding paramters for CT algorithm.
- **set_ct_encode_inline(arg); -** Specify whether to inline the methods for creating the watermark at the call point; set to true/false
- **set_ct_encode_replace(arg); -** Allow sandmark to automatically choose a class that best represents the watermark graph node; set to true/false
- **set_ct_dumpIR(arg); -** Print out the intermediate code; set to true/false
- **set_min_wmparts(numparts); -** Specify number of parts the watermark is to be broken.
- **set_encode_consts(arg); -** Encode the watermark wither as constants in the opaque predicate or using the rank; set to true/false
- **set_use_opaque_methods(arg); -** Encode the watermark either as an inserted opaque method or as an inserted opaque predicate; set to true/false
- **set_reuse_methods(arg); -** Reuse opaque methods when the rank is used to encode the value of watermark; set to true/false
- **setPrefix(prefixStr); -** The prefix of the string in which we hide the ConstantString watermark.
- **exit(); -** Exit SandMark shell.

Additional options:

- **runEmbed(jarString); -** Embed a watemark on multiple input jar files; Specify jarString in the format <jar1:jar2: ... :jarN>
- **runRecognizeWM(jarString); -** Run recognition on multiple jar files; Specify jarString in the format <jar1:jar2: ... : jarN>
- **embedMultipleWM(algoWmarkStr); -** Run multiple watermark algorithm embed operations on an input jar file. The output of the first embedding is fed as the input to the next embedding in sequence. Specify algoWmarkStr in the format <algo1.wmark1>:<algo2.wmark2>: ... :<algoN.wmarkN>
- **recognizeMultipleWM(algoStr); -** Run multiple algorithm recognition on a single input jar file; Specify algoStr in the format <algo1:algo2: ... :algoN>

---

## Sample_Scripts for SandMark CLI

- Script1-Statically Embedding a watermark
- Script2-Dynamically Embedding a watermark
- Script3-Obfuscation
- Script4-Running a recognition

---

```
#   setenv sandmarkCLI  "java -classpath sandmark.jar sandmark.smash.SandmarkCLI"
```

- **Statically embedding a watermark**

  $sandmarkCLI -A AddMethField -i TTT.jar -o outputTTT.jar -w 1234 -E -k 1111
  > Embed 1234  in TTT.jar using the static watermark algorithm AddMethField and key value 1111;
  outputTTT.jar is the watermarked jar file.

- **Dynamically embedding a watermark**

  $sandmarkCLI -A CT -i TTT.jar -o TTT.tra -m TTTApplication -T
  > Create a trace file TTT.tra from the input file TTT.jar; TTTApplication is the main class specified
  $sandmarkCLI -A CT -i TTT.jar -t TTT.tra -w 12345 -E
  > Embed the wateramark 12345 in TTT.jar using the tracefile TTT.tra. The default watermaked file is TTT_wm.jar

- **Obfuscation**

  $sandmarkCLI -A Method2RMadness -i TTT.jar   -O
  >  Run the Method2RMadness obfuscation on TTT.jar ; TTT_obf.jar is the default obfuscated jar file created.

- **Running a recognition**

```
$sandmarkCLI -A AddMethField -i outputTTT.jar -k 1111 -R
```

> Run the recognition for AddMethField algorithm on outputTTT.jar file with key value 1111.

---

## Sample_Scripts for SandMark shell

- Script1-Statically Embedding a watermark
- Script2-Dynamically Embedding a watemark
- Script3-Obfuscation
- Script4-Run multiple watermark algorithms on a jar file in sequence
- Script5-Run a recognition

---

```
# java -classpath sandmark.jar sandmark.smash.Smash
```

> Initialize and start sandmark shell. A **smash$** prompt will be displayed.

- **Statically Embedding a watermark**

```
> selectAlg("ConstantString");
 > setInputFile("TTT.jar");
 > setKey("9876");
 > setWM("12345");
 > setPrefix("sm$");
 >embed();
```

- **Dynamically Embedding a watermark**

```
> selectAlg("DynamicAA");
 > setInputFile("TTT.jar");
 > setTraceFile("TTT.tra");
 > setMainClass("TTTApplication");
 > startTrace();
 > set_min_wparts("2");
 > set_encode_as_constants("true");
 > set_use_opaque_methods("true");
 > set_reuse_methods("false");
 >embed();
```

- **Obfuscation**

```
> selectAlg("Buggy Code");
> setInputFile("Ackermann.jar");
> setOutputFile("AckermannOutput.jar");
> obfuscate();
```

- **Run multiple watermark algorithms on jar file in sequence**

```
> setInputFile("TTT.jar");
 > embedMultipleWM("AddMethField.1234:BogusExpression.56789:HatTrick.4444");
```

- **Run a recognition**

```
> setInputFile("TTT_wm.jar");
 > selectAlg("BogusInitializer");
 > setKey("1111");
 > recognize();
```