

SMAC outputs a variety of information to log files, trajectory files, and state files. Most of the files are human readable, and this section describes these files.

NOTE: All output is written to the **outdir** in the **--runGroupName** sub-directory.

0.1 Logging Output

SMAC uses slf4j (<http://www.slf4j.org/>), a library that allows for abstracting and replacing the logging system with ease, and uses logback (<http://logback.qos.ch/>) as the default logging system. While there is limited ability to change logging options via the command line (e.g., **--logLevel**, **--consoleLogLevel**, **--logAllCallStrings**, **--logAllProcessOutput**), one can edit `conf/logback.xml`, to get much more control over the logging of SMAC. For more details of how to edit this file consult the logback documentation.

NOTE: If you replace the logger in SMAC or modify the configuration file, the logging command line options may no longer work.

By default SMAC writes the following logging files out to disk (NOTE: The N represents the **--numRun** setting):

log-run N .txt A log file that contains a full dump of all the information logged, and where it was logged from.

log-warn N .txt Contains the same information as the above file, except only from warning and higher level messages.

log-err N .txt Contains the same information as the above file, except only from error messages.

runhashes-run N .txt A file that contains only the Run Hash Codes for a given run see the corresponding entry in the **FAQ**.

0.1.1 Interpreting the Log File

SMAC basically goes through three phases when executing:

- Setup Phase Input files are read, and their arguments validated. Everything necessary to execute the Automatic Configuration Phase is constructed. This phase ends (barring anything that must be lazily loaded), once the message `Automatic Configurator Started` is logged.
- Automatic Configuration Phase: SMAC is now actively configuring the target algorithm. SMAC will spend most of it's time here, and outputs it's progress. The most important output is the Runtime Statistics which will appear like:

```
[INFO ] *****Runtime Statistics*****
Iteration: 35
Incumbent ID: 64 (0x18824F)
Number of Runs for Incumbent: 70
Number of Instances for Incumbent: 70
Number of Configurations Run: 67
Performance of the Incumbent: 1589.1414639125514
Total Number of runs performed: 242
Wallclock time: 18.213 s
Wallclock time remaining: 2.147483628787E9 s
Configuration time budget used: 84056.83939320213 s
Configuration time budget remaining: 2343.160606797872 s
Sum of Target Algorithm Execution Times \
(treating minimum value as 0.1): 84036.36939320213 s
```

```

CPU time of Configurator: 20.47 s
User time of Configurator: 19.6 s
Total Reported Algorithm Runtime: 84033.27806288192 s
Sum of Measured Wallclock Runtime: 0.0 s
Max Memory: 3505.8125 MB
Total Java Memory: 1249.0625 MB
Free Java Memory: 719.8940582275391 MB
[INFO ] *****

```

While most of the fields are self-explanatory some deserve special attention:

Incumbent ID

The second ID (0x18824F) is a hex-code that represents the configuration anywhere / everywhere it is logged. The first ID, 64, occurs in context where we know the configuration is intended to be run. This ID will correspond to the ID in the state files. The second ID will always associate with a unique first ID, but not conversely. The second ID roughly represents the specific configuration in memory ¹.

Performance of the Incumbent

This represents the performance of the incumbent under the given **run_obj** and **overall_obj** on the runs so far.

Configuration time budget used

The tuner time that has been used so far.

Sum of Target Algorithm Execution Times

This represents the contribution of the algorithm runs to the Tuner Time (if applicable), in general each run contributes the minimum of 0.1 and it's reported runtime. This parameter differs from Sum of Measurement Wallclock Runtime in that the latter is a direct sum. If you are only running on algorithms with large runtime, this difference may be 0.

- Validation Phase, depending on the options used this can also take a large fraction of SMAC's runtime. The logic here is actually quite simple, as it largely only requires running many algorithm runs and computing the objectives from them.

At the end of Validation the Runtime Statistics (from the Automatic Configuration Phase) are displayed again, as is the following information

1. The performance of the incumbent on both the training and test set.
2. A sample call of the final incumbent (selected configuration)
3. The complete configuration selected (without inactive conditionals)
4. The complete configuration selected (with inactive conditionals)
5. The Return value of SMAC (generally 0 if successful)

¹Specifically every time a configuration is modified, this number is incremented. In cases where the configuration space is small, or we are examining a small part of it, SMAC may end up back at the same configuration again. As far as the behaviour of SMAC is concerned these are identical, the ID is only ever used for logging.

0.2 State Files

State files allow you to examine and potentially restore the state of SMAC at a specific point of its execution. The files are written to the `state-run N /` sub-directory, where N is the value of `--numRun` option.

All files have the following convention as a suffix either `it` or `CRASH` followed by either the iteration number M , or in some cases `quick` or `quick-bak`.

The state is saved for every iteration m , where $m = 2^n$ $n \in \mathbb{N}$, additionally it is saved when SMAC completes whether successfully or due to crash.

The following files are saved in this state directory (ignoring the suffix):

java_obj_dump Stores (Java) serialized versions of the the incumbent and the random object state. In general there is no need to look at this file, and it is not human readable.

paramstrings Stores a human readable setting of each configuration ran, with a prefix of the numeric id of the configuration (as used in the logs, and other state files).

uniq_configurations Stores the configurations ran in a more concise but effectively un-human readable form. The first column again is the numeric id of the configuration (as used in the logs, and other state files).

run_and_results Stores the result of every run of the target algorithm that SMAC has done. The first 13 columns (after the header row are designed to be backwards compatible with SMAC versions 1.xx. Each column is labelled with what data it contains, the following columns deserve some description.

Instance ID This is the instance used, and is the n^{th} **Instance Name** specified in the **instance_file** option.

Response Value(y) This is the value determined by the **run_obj** on the run.

Censored Indicates whether the Cutoff Time Used field is less than the **cutoff_time** in the original run. 0 means false, 1 means true.

Run Result Code This is a mapping from the Run Result to an integer for use with previous versions.

param-file If `--saveContext` is enabled, a copy of the **paramfile** will be in the state folder

instances If `--saveContext` is enabled, a copy of the **instance_file** will be in the state folder

instance-features If `--saveContext` is enabled, and SMAC is running with features, then a copy of the **feature_file** will be in the state folder.

scenario If `--saveContext` is enabled, and SMAC is using a scenario file, then a copy of the **--scenarioFile** will be in the state folder.

0.3 Trajectory File

SMAC also outputs a trajectory file into identical files `traj-run- N .txt`² and `traj-run- N .csv`. These files outline the incumbent (by id) over the course of execution and its performance. The first line gives the `--runGroupName`, and then the `--numRun`.

The rest of the file follows the following format:

²This file is outputted for backwards compatibility with existing scripts.

Column Name	Description
Total Time	Sum of all execution times and CPU time of SMAC
Incumbents Mean Performance	Performance of the Incumbent under the given -runObjective and -overallObjective
Incumbent's Performance σ	Outputs -1 Currently
Incumbent ID	The ID of the incumbent as listed in the param_strings file 0.2.
acTime	CPU Time of SMAC
Remaining Columns	Give a name value mapping for the configuration value as given by the <code>Incumbent ID</code> column

0.4 Validation Output

When Validation is completed four files are outputted, (again N is the value of the **-numRun** argument):

1. `rawValidationExecutionResults-runN.csv`:

CSV File containing a list of the configuration, seeds & instance run and the corresponding result and the result of the target algorithm execution. This file is mainly for debugging.

2. `validationInstanceSeedResult-runN.csv`:

CSV File containing a list of seeds & instances and the resulting response value. Again this file is mainly for debugging, but is easier to parse than the previous.

3. `validationResultsMatrix-runN.csv`:

CSV File containing the list of instances on each line, the next column is the aggregation of the remaining columns under the **overall_obj**. Finally there is one additional row that gives the aggregation of all the individual **overall_obj**, aggregated in the same way.

`classicValidationResults-runN.csv`

CSV File containing the result of the validation. The columns are defined as follows:

Column Name	Description
Tuner Time	The tuner time when validation occurred
Emperical Performance	The incumbent's performance on the training set
Test Set Performance	The incumbent's performance on the test set
AC Overhead Time	Total CPU Time Used by the Automatic Configurator