



# **Intel® SoC Watch User Guide for Google Android\* and Linux\* OS**

September 2024

Version 2024.6

Intel Corporation

[www.intel.com](http://www.intel.com)

[Legal Information](#)

---

# Contents

<b>Legal Information .....</b>	<b>3</b>
<b>Version History .....</b>	<b>4</b>
<b>Chapter 1: About Intel® SoC Watch</b>	
Related Information .....	7
<b>Chapter 2: Installation</b>	
<b>Chapter 3: Getting Started with Intel® SoC Watch</b>	
Collect on Linux* OS.....	9
Set Up Collection on Linux.....	9
Collection on Linux or ChromeOS .....	9
Collect on Android OS .....	9
<b>Chapter 4: Options Quick Reference</b>	
General Options .....	11
Post-processing Options .....	11
Collection Options .....	12
Feature Names (Individual) .....	14
Feature Group Names.....	16
<b>Chapter 5: Viewing Intel SoC Watch Results with Intel® VTune™ Profiler</b>	

# Legal Information

---

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Performance varies by use, configuration and other factors. Learn more at [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex).

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel, the Intel logo, Intel Atom, Intel Core, Intel Xeon Phi, VTune and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

© **Intel Corporation**

---

# Version History

---

These are the main releases of Intel® SoC Watch:

Date	Revision	Description
June, 2019	2.11	Improves handling of unrecognized CPUs, reporting S-state when hibernation occurs, and other bug fixes.
September, 2019	2019.12	Added support for Intel platform code named Ice Lake. Modified hw-cpu-pstate reporting.
October, 2019	2019.13	Fixed issue in hw-cpu-pstate for Intel platform code named Ice Lake.
November, 2019	2020.1	Added support for Intel platform code named Comet Lake.
February, 2020	2020.2	Added collection of tool usage analytics. Added new features pch-slps0, pch-slps0-dbg. Improved error messages and help output. Enhanced driver security.
June, 2020	2020.3	Bug fix release.
July, 2020	2020.3.1	Bug fixes
September, 2020	2020.3.2	Bug fixes .
October, 2020	2020.4	Added support for Intel platform code named Tiger Lake. Added support dgfx-pwr support for discrete graphics card code named DG1. Added non-root user support. Added topology label in reports for some metrics. Re-named feature cpu-gpu-concurrency to cpu-igpu-concurrency. Removed support for older platforms.
November, 2020	2020.5	Changed hw-cpu-pstate to report frequencies per thread rather than per core. Added term <i>integrated</i> to hw-igfx-cstate and hw-igfx-pstate report titles. Fix for hw-cpu-cstate reporting of core concurrency.
April, 2021	2021.1	Added system name, OS name, and Intel PMT GUID to output reports. Included throt-rsn and other sampled count results in Automation_Summary. Re-ordered feature reports in summary output. Updated socwatch driver to build for Linux 5.12.
July, 2021	2021.2	Bug fix release.
September, 2021	2021.3	Added PMT-based metric support when using Linux kernel v5.11.
October, 2021	2021.3.1	Improved support for platforms with many sockets and/or discrete graphics cards. Modified option --update-usage-consent.
November, 2021	2021.4	Added support for Intel platforms code named Rocket Lake and Tiger Lake - H.

Date	Revision	Description
		Added support for Intel platforms code named Ice Lake -X and Cooper Lake -X. Bug fixes.
February, 2022	2022.1	Added support for Intel platform code named Alder Lake and Intel® Arc Discrete Graphics Card . Bug fixes.
April, 2022	2022.2	Bug fixes.
June, 2022	2022.3	Bug fixes.
August, 2022	2022.4	Bug fixes.
September, 2022	2022.5	Bug Fixes.
November, 2022	2022.6	Report complete topology path on hybrid systems. Bug fixes for data correctness.
January, 2023	2023.0	Deprecation of Google Analytics support. Bug fixes for data correctness.
February, 2023	2023.1	Bug fixes for data correctness.
April, 2023	2023.2	Added support for Intel platform code named Raptor Lake. Added support for Intel server platform code named Sapphire Rapids -Xeon. Bug fixes.
June, 2023	2023.3	Bug fixes.
August, 2023	2023.4	Bug fixes
October, 2023	2023.5	Change in scope for certain group features including -f sys.
November, 2023	2023.6	Added support for Intel server platform code named Emerald Rapids-Xeon. Bug Fixes
December, 2023	2023.7	Bug fixes.
January, 2024	2024.0	Enabled additional features support on Intel platform code named Meteor Lake.
February, 2024	2024.1	Bug fixes.
April, 2024	2024.2	Deprecation of support for older atom based features. Bug Fixes
May, 2024	2024.3	Enabled support for Intel discrete graphics code named Ponte Vecchio. Enabled support for the feature hw-cpu-hwp on Intel server platforms. Bug Fixes
July, 2024	2024.4	Bug fixes including data correctness issue in the feature sata-lpm.
September, 2024	2024.6	Added support to Intel platforms code named Elkhart Lake, Arrow Lake and Lunar Lake.

---

Date	Revision	Description
		Bug Fixes

# About Intel® SoC Watch

# 1

Intel® SoC Watch is a command line tool for monitoring and debugging system behaviors related to power consumption on Intel® architecture-based platforms. It reports active and low power states for the system/CPU/GPU/devices, processor frequencies and throttling reasons, wakeups, and other metrics that provide insight into the system's energy efficiency. The tool includes utility functions that include delaying the start of collection and launching an application prior to starting collection.

Data is collected from both hardware and OS sources. When using the default mode of collection, the tool collects data at normally occurring OS context-switch points so that the tool itself is not perturbing the system sleep states. Tool overhead when collecting during idle scenarios can be < 1%, however active workloads with a high-rate of context switching will increase the overhead. A minimum collection interval is used to control the rate of collection.

Intel SoC Watch writes a summary report file (.csv) at the end of collection on the system under analysis (target system), allowing immediate access to results. Additional result files can be specified including: an import file (.pwr) for Intel® VTune™ Profiler that can be used for visualization of correlated timelines for all the collected metrics with powerful zoom and filtering functions, and a time trace file (.csv) that can be viewed as a timelines in tools like Microsoft\* Excel\*.

## Related Information

See the Intel® SoC Watch Release Notes for information on new features as well as known issues.

For online help, including information about importing results into Intel® VTune™ Profiler, see the Energy Analysis User Guide (<https://software.intel.com/en-us/energy-analysis-user-guide>).

### Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

## ***Installation***

---

See the *Intel® SoC Watch Release Notes* for supported platforms and installation instructions.

# Getting Started with Intel® SoC Watch

## 3

The following steps assume the Intel SoC Watch drivers and executables are installed. See the *Intel SoC Watch Release Notes* for instructions on how to install Intel SoC Watch.

Use the following steps to quickly collect processor C-state and P-state data for 60 seconds and import it into Intel VTune Profiler for analysis.

## Collect on Linux\* OS

### Set Up Collection on Linux

To collect directly on a Linux target, login as 'root'. To collect remotely on a target device running Linux, use ssh to login to your target device as root.

### Collection on Linux or ChromeOS

1. Navigate to the Intel SoC Watch directory:

```
cd <path>/socwatch/socwatch_linux_[version]
```

2. Setup the collection environment:

```
source ./setup_socwatch_env.sh
```

3. Build the Intel SoC Watch driver:

```
sudo -E ./build_drivers.sh -l -n
```

4. Install the Intel SoC Watch driver:

```
sudo insmod drivers/socwatch2_13.ko
```

Some metrics may not be collected if older driver version is used.

5. Create a results directory:

```
mkdir results
```

6. Collect data.

For example, this command generates the test.csv, test.sw2 and test.pwr files in the results directory.

```
./socwatch -r vtune -m -f cpu-cstate -f cpu-pstate -t 60 -o ./results/test
```

7. View the summary results.

```
cat ./results/test.csv
```

8. To view results in Intel VTune Profiler on your host system, copy the test.pwr file from the target to the host using scp. The following step assumes a Windows host.

```
scp root@<your_target_IP>:<path>/socwatch/<path>/results/test.pwr c:\results\.
```

## Collect on Android OS

1. On the host system, establish a root adb shell on the target:

```
adb root
adb shell
```

2. Load the Intel SoC Watch driver:

```
insmod <path_to_socwatch_driver>/socwatch2_13.ko
```

**3. Confirm the drivers are loaded:**

```
lsmod
```

Confirm the loaded drivers are included in the list of installed modules.

**4. Setup the collection environment. This step assumes the default install directory was used.**

```
cd /data/socwatch  
source ./setup_socwatch_env.sh
```

**5. Collect data and generate the test.csv, test.sw2, and test.pwr files in the results directory. This step assumes the /data/socwatch/results directory exists.**

```
./socwatch -r vtune -m -f cpu-cstate -f cpu-pstate -t 60 -o ./results/test
```

**6. Exit the adb shell:**

```
exit
```

**7. Use adb to pull the result files to the host:**

```
adb pull /data/socwatch/results/test.csv c:\results  
adb pull /data/socwatch/results/test.sw2 c:\results  
adb pull /data/socwatch/results/test.pwr c:\results
```

## 4

# Options Quick Reference

Invoke Intel SoC Watch with root privilege, using the following syntax:

```
socwatch <general options><post-processing options><collection options>
```

- Order of options does not matter unless specifically noted.
- Help is displayed if no option is specified.
- All features are not available on all systems, so the help text is dynamic, meaning it displays only the collection options that are supported by the system on which it is run. The metrics available differ because of changes in the system's hardware architecture support. This User's Guide contains a list of all metrics across all systems.
- You can specify feature names that are not available or not enabled on a particular system. When the tool starts, it will display console messages regarding features that cannot be collected, but collection will proceed if at least one feature is valid on that system.

Intel SoC Watch terminates data collection for one of three reasons (whichever occurs first):

1. the `--time` option was specified and the timer elapsed,
2. the `--program` option was used and the specified program exited,
3. a Ctrl-C interrupt was entered in the command window.

The location and name of the results files is displayed at the end of a collection. The summary report will be there with that name and a `.csv` extension. Raw data files and additional files based on post-processing options specified on the command line are located there as well, all with the same base name (default name is `SoCWatchOutput`).

## NOTE

Result files are replaced if the same name is used for multiple collections.

## General Options

The following options display information about the tool or system on which it is run.

Abbreviation	Option Name	Description
	<code>--export-help</code>	Write help output to JSON formatted file.
<code>-h</code>	<code>--help</code>	Display tool usage information and exit. The help shown is specific to the system on which it is run. Only metrics supported by the system architecture will be listed.
<code>-l</code>	<code>--log &lt;filename&gt;</code>	Redirect all console output, including errors, to specified file.
	<code>--print-fms</code>	Display CPU ID as Family.Model.Stepping and exit.
<code>-v</code>	<code>--version</code>	Display tool version information and exit.

## Post-processing Options

The following options affect how results are reported and where they are stored.

Abbreviation	Option Name	Description
-i	--input <filename>	Specify the collection name (with full path) of an existing collection to generate reports.
-o	--output <filename>	Specify the output collection name (default "SoCWatchOutput"). Specifying <code>console</code> as the filename will cause the summary results to also write to stdout. If the filename already exists, the previous results will be replaced.
-r	--result <result_type>	<p>Specify the type of result to generate. This option can be repeated to get multiple types of reports. Following are the result types that can be specified:</p> <ul style="list-style-type: none"> <li>• <code>sum</code> Write summary reports to .csv file. [default]</li> <li>• <code>int</code> Write over-time data to _trace.csv file.</li> <li>• <code>vtune</code> Generate .pwr file for import to Intel VTune Profiler.</li> <li>• <code>json</code> Generate .swjson file.</li> <li>• <code>auto</code> Write summary results as a single line to file Automation_Summary.csv in current directory. Appends results, does not overwrite. If column headers for the new result changed, new headers will be inserted. Use to generate sets of data in a single file for comparison.</li> </ul>

## Collection Options

These options affect what is collected and how it is collected.

Abbreviation	Option Name	Description
-f	--feature <name>	Specify which metric to collect, choose from the group names or individual names listed in the tables below. This option can be repeated to collect multiple metrics in a single run. Most features can be collected simultaneously, exceptions noted in the table of feature names.
-m	--max-detail	<p>Collect all data available for each feature specified. This will cause snapshot metrics to be sampled. Use of this option can increase tool overhead, so best used only when timeline of the data is needed or when collecting across system entry to hibernation.</p> <p>Without this option, the tool collects data at the minimum required by the data source for best accuracy.</p> <p>Data may be traced, sampled, or snapshot.</p> <ul style="list-style-type: none"> <li>• Traced data is obtained at state transition points resulting in accurate summary and timeline results.</li> </ul>

Abbreviation	Option Name	Description
		<ul style="list-style-type: none"> <li>Sampled data is read at OS context switch points (or at timed intervals if polling option is used). This is less accurate as changes that take place between samples will not be measured. Metrics that come from hardware status/ state data must be sampled.</li> <li>Snapshot data can be read at the beginning and end of the collection and the difference gives an accurate result with lowest overhead, but no timeline. Only metrics that come from hardware accumulators can be snapshot.</li> </ul> <p>The algorithm used to determine the collection method for each data type is as follows:</p> <p>If <code>-m</code> is specified:</p> <p style="padding-left: 40px;">if the data can be traced, trace it; else sample it.</p> <p>If <code>-m</code> is not specified:</p> <p style="padding-left: 40px;">if the data can be snapshot, snapshot it; else if the data can be traced, trace it; else sample it.</p>
<code>-n</code>	<code>--interval &lt;milliseconds&gt;</code>	<p>Specify the time in milliseconds that should pass before reading next hardware data sample (default 100 ms). For default collection mode, this is the minimum time between sampling at context switch points. When <code>--polling</code> option is used, this is actual time between samples.</p> <p>The minimum polling interval is 1ms. However, using low polling intervals will result in higher overhead and may fail to measure some metrics (e.g. bandwidths) with intervals shorter than the default.</p>
	<code>--no-post-processing</code>	Do not generate the summary file or other result files at the end of collection. Use <code>-i</code> option to process the intermediate results files and generate summary or other result file types at a later time.
	<code>--polling</code>	Make data collection occur at regular intervals rather than at context switch points. Use the <code>--interval</code> option to set the interval period (default: 100ms). Use of this option significantly increases perturbation of sleep states because it employs a timer which will interrupt sleep states, increase wakeup counts, and change timer resolution.

Abbreviation	Option Name	Description
-p	<code>--program &lt;application&gt; &lt;parameters&gt;</code>	Specify the name of an executable to be started automatically prior to collection. The name can be followed by zero or more arguments that will be passed to the program.  <b>NOTE</b> This option must occur at the end of the command line, everything following the executable name will be given to it as arguments.
	<code>--program-delay &lt;seconds&gt;</code>	Specify number of seconds to wait before starting the program specified by -p. Has no effect if -p not used.
-s	<code>--startdelay &lt;seconds&gt;</code>	Specify number of seconds to wait before starting collection of data.  If used with -p and --program-delay, this delay is applied after the program starts.
-t	<code>--time &lt;seconds&gt;</code>	Specify collection duration in seconds. Collection will stop when this time has elapsed unless Ctrl-C is entered or an executable specified with --program option exits prior to the specified duration.
-z		Automatically enter Suspend for the duration of the collection. Will automatically exit Suspend when the -t specified time expires. If system is woken from Suspend prior to the end of the duration, the collection will stop as well. If --start-delay is specified, it occurs prior to entering Suspend.

## Feature Names (Individual)

The available feature names for the `--feature` option and their collection methods are listed below. You can specify multiple feature names individually or using group names described in the Feature Group Names section.

Note that every feature listed is not available on every platform supported by Intel SoC Watch. The `--help` option is dynamic, only showing features available for the platform on which it is run. Use it to determine which features are supported. You can specify unsupported features on the command line and the tool will simply display a message for those that cannot be collected, but continue with collection if there is at least one that is supported.

Collection methods are indicative of a metric's level of accuracy and overhead. Traced collection provides high accuracy along with precise transition points between states. Sampled collection is least accurate since transitions can occur which are never noted. Sampled data needs to be read at intervals throughout the collection period which increases tool overhead. Increasing the sampling rate (reading at closer intervals) will improve accuracy but increase overhead. Snapshot collection means the data comes from an accumulator so it can be collected only at the start and end of the collection period and give perfect accuracy. This gives accuracy and the lowest overhead. If the `--max-detail (-m)` option is given, the Snapshot metrics will instead be read at the same intervals as the Sampled metrics throughout the collection, so that you can generate a trace file to see how it changed overtime.

Name	Collection Methods	Description
cdie-ccp-llc-bw	Sampled	Converged Core Perimeter (per Core/Module) to LLC Read and Write bandwidth for Compute Die. [from PMT]
cdie-cstate-dbg	Snapshot	Reasons blocking entry to lower-power Die C-states and wakeup from Die C-states for Compute Die. [from PMT]
cdie-cstate-res	Snapshot	Residency for Die C-states (DCx.y) for Compute Die. [from PMT]
core-temp	Sampled	IA core temperature statistics, from hardware status data.
core-volt	Sampled	Calculate core voltage, from hardware status data. This data can only be collected on Intel Atom <sup>®</sup> Processor-based SoCs for systems code named Apollo Lake.
cpu-igpu-concurrency	Snapshot	Concurrent active time of CPU and integrated GPU, from hardware accumulators.
cpu-pkgc-dbg	Snapshot	Counts for reasons blocking entry to Package C-states and wakeups from Package C-states, from hardware accumulators.  <b>NOTE</b> This data is retrieved from PMT on Intel Platforms starting from Meteor Lake.
cpu-pkg-cstate-res	Snapshot	Residency for Package C-states (PCx). [from PMT]
ddr-bw	Sampled	Total bandwidth per memory channel, from hardware accumulators.  The hardware accumulator data is always collected over time due to frequent overflow, so snapshot is not available.
dram-pwr	Sampled	Total DRAM power consumption from hardware accumulators.  This data can be collected on Intel server platforms.
hw-cpu-pstate	Sampled	CPU P-state operating frequency residencies, from trace data.
hw-igfx-cstate	Snapshot*	Integrated GPU C-state residency (RC6), from hardware accumulators. *Always sampled due to short overflow time period.
hw-igfx-pstate	Sampled	Integrated GPU P-states approximated residency, from hardware status data.
ia-throt-rsn	Sampled	Reasons for throttling the CPU frequency, from hardware status data.

Name	Collection Methods	Description
igfx-throt-rsn	Sampled	Reasons for throttling the integrated GPU frequency, from hardware status data.
io-bw	Sampled	Total IO bandwidth Reads and Writes. [from PMT]
npu-bw	Sampled	NPU memory bandwidth. [from PMT]
npu-dstate-cnt	Sampled	Entrance count for NPU D-states (D0/D0ix). [from PMT]
npu-dstate-res	Sampled	Residency for NPU D-states (D0/D0ix). [from PMT]
pch-slps0	Snapshot	PCH SLP_S0 residency, from hardware accumulator.
pch-slps0-dbg	Sampled	Blocking reasons for SLP_S0, from hardware status data.
pkg-pwr	Snapshot	Total SoC/Package power consumption, from hardware accumulator.
psys-pwr	Snapshot	PSYS power statistics. [from PMT]
ring-throt-rsn	Sampled	Reasons for throttling the ring clock frequency, from hardware status data.
vccgt-pwr	Snapshot	VCCGT power statistics. [from PMT]

## Feature Group Names

The following features are groupings of the previously described features. These group names can be used to simplify command lines to collect multiple features concurrently. For example, `-f cpu` can replace the `-f cpu-cstate -f cpu-pstate` in a command line.

If a group includes a feature that is not enabled on the target platform, that feature will be ignored and collection continue, as long as there is one feature that can be collected.

All features are not supported on all platforms, a group will only include the supported features. Use the `--help` option on the target platform to see the list of group names and specific features included each group.

Name	Description
chipset-all	All PCH related metrics (all PCH active groups)
cpu	cpu-hw
cpu-hw	Most CPU metrics obtained from hardware data sources
device	Device state residency metrics
gfx	All graphics metrics from hardware and OS. gfx-hw
gfx-hw	Most GPU metrics obtained from hardware data sources.
power	Power/energy metrics
sys	Spectrum of metrics commonly used to get general information about platform power behavior. Provides information about physical components (CPU, GPU) and not include firmware component information

Name	Description
temp	Temperature metrics
throt	Frequency throttling reason metric.

## ***Viewing Intel SoC Watch Results with Intel<sup>®</sup> VTune<sup>™</sup> Profiler***



You can analyze Intel SoC Watch data graphically using the Intel<sup>®</sup> VTune<sup>™</sup> Profiler GUI. Intel<sup>®</sup> VTune<sup>™</sup> Profiler provides a dynamic timeline view for interacting with Intel SoC Watch data and provides powerful filtering of data for in-depth analysis of a platform's power management behavior.

For detailed instructions, refer to the [Energy Analysis](#) section of the Intel<sup>®</sup> VTune<sup>™</sup> Profiler Help.