



# Intel® SoC Watch User Guide for Google Android\*, Google Chrome\*, Linux\* and QNX RTOS\* OS

**(for use under NDA only)**

4th November 2019

Version 2020.1

Intel Corporation

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

[Legal Information](#)

**Intel Confidential**

---

# Contents

<b>Legal Information .....</b>	<b>3</b>
<b>Revision History.....</b>	<b>4</b>
 <b>Chapter 1: About Intel® SoC Watch</b>	
Intended Audience.....	6
Related Information .....	6
 <b>Chapter 2: Installation</b>	
 <b>Chapter 3: Getting Started with Intel® SoC Watch</b>	
Preparing the Target on Chrome* OS .....	8
Preparing the Target on QNX* RTOS.....	8
Collect on Linux* or Chrome* OS .....	9
Set Up Collection on Linux.....	9
Set Up Collection on Chrome OS .....	9
Collection on Linux or Chrome OS .....	9
Collect on Android OS .....	10
Collect on QNX RTOS .....	10
 <b>Chapter 4: Options Quick Reference</b>	
General Options .....	12
Post-processing Options .....	12
Collection Options .....	13
Feature Names (Individual) .....	15
Feature Group Names.....	26
 <b>Chapter 5: Viewing Intel SoC Watch Results with Intel®           VTune™ Amplifier</b>	

# Legal Information

---

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

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

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.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications. Current characterized errata are available on request.

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.

\*Other names and brands may be claimed as the property of others.

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.

## **Copyright 2013-2019 Intel Corporation.**

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you (**License**). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

---

# Revision History

---

Revision Number	Description	Revision Date
2.0	Initial release for 2.0 product.	August 2015
2.1	Updated Building the Kernel Modules section	November 2015
2.1.1	Updated for 2.1.1 product release	December 2015
2.2	Updated for 2.2 product release and to combine Android*, Chrome*, Linux*, and OS X* User's Guides	June 2016
2.2.1	Updated for 2.2.1 product release Added NDA support for OSX Update from socperf1_2.ko to socperf2_0.ko Bug fixes	June 2016
2.3.0	Updated for 2.3.0 product release	October 2016
2.3.1	Updated for 2.3.1 product release	April 2017
2.4	Update aligning available metrics and common reporting across all OSes, including Windows OS for the first time. Includes many new metrics for Intel® Core™ platforms.	November 2017
2.4.1	Bug fix release. No change in this document.	November 2017
2.5	Added support for QNX*, added feature pch-ip-lat-limit and added Intel VTune Amplifier import support for most PCH metrics.	February 2018
2.6	Enhancements include initial support for Intel platform code named Ice Lake, added new s0i3-sstate-dbg metric, and improved PCIe LPM and PCH reporting.	April 2018
2.6.1	Adds feature xhci-lpm. Also adds warning if too many telemetry-based metrics requested, corrects minimum LTR reporting for platform LTR.	May 2018
2.7	Added average frequency report, new options (program-delay), new metric (cpu-pkgc-cfg), new group names, support for Intel platform code named Amber Lake, support for S0ix substates, fixed issues in ddr-bw, automation summary, multiple pkg handling, PCH and XHCI reporting among others.	August 2018
2.8	Added new tables to cpu-pkgc-dbg. Fixed errors in pch-slps0 and hw-cpu-cstate PC10 reporting (requires kernel version 4.4 or newer).	November 2018
2.9	Added tcss-state, trace file report grouping, and informational messages. Fixes for PCH metrics, xhci-lpm, and more.	January 2019

Revision Number	Description	Revision Date
2.10	Added support for Intel platform code named Comet Lake, added feature pmc-ip-pg-res for Intel Atom platforms, added histogram for CPU frequency, added TypeC subsystem metric tcss-cfg-status, and bug fixes.	March 2019
2.11	<p>Added option (-z) to automatically put the system into suspend state during a collection.</p> <p>Added support for Intel platform code named Tiger Lake and limited support for Intel platform code named Ice Lake-Xeon.</p> <p>Added reporting of power limits (PL1 and PL2) to feature -f pkg-pwr on Intel platform code named Apollo Lake.</p> <p>Includes support for Intel platform code named Cascade Lake-Xeon.</p> <p>Improves handling of unrecognized CPUs, reporting S-state when hibernation occurs, and other bug fixes.</p>	June 2019
2019.12	<p>Added support for Intel PCH code named Comet Lake V.</p> <p>Added feature s0ix-subs-res.</p> <p>Improved sata-lpm DevSlp Capabilities reporting, cpu-pkgc-dbg, and PCH metrics for various platforms. Modified pch-ip-active percentage calculation.</p> <p>Modified hw-cpu-pstate reporting.</p>	September 2019
2019.13	<p>Added feature s0ix-subs-status for Intel platform code named Tiger Lake.</p> <p>Fixed issue in ddr-bw and hw-cpu-cstate for Intel platform code named Tiger Lake.</p> <p>Fixed issue in hw-cpu-pstate for Intel platform code named Ice Lake.</p>	October 2019
2020.1	<p>Added support for Intel platform code named Elkhart Lake and Intel PCH code named Mule Creek Canyon.</p> <p>Added support for Intel platform code named Tiger Lake-H.</p> <p>Fixed issues in pcie-lpm, pcie-ltr among others.</p>	November 2019

# About Intel® SoC Watch

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, IO subsystem low power states and latency tolerance, PCH activity, memory and display refresh states, 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™ Amplifier 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\*.

## Intended Audience

Use this document if you use Intel SoC Watch to analyze power consumption on a Linux, Android, Chrome, or QNX system, or a system running the ARC\* hypervisor.

## 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™ Amplifier, 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

For QNX\* Real Time Operating System (RTOS), the following steps assume that the Intel SoC Watch executables are installed.

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 Amplifier for analysis.

## Preparing the Target on Chrome\* OS

The following section applies only to target devices running Google\* Chrome OS. After rebooting the system, complete the following steps on the target device once before running Intel SoC Watch collections:

1. Target: `cd /`
2. Target: `find . -name module_locking`
3. Target: `sudo echo 0 > </path_to_module_locking_file found in previous step>`

If module\_locking is set to 1, this step is required to disable module\_locking.

4. Target: `sudo mount -o remount,rw /home`

Change read only bit to read write on /home partition.

5. Target: `sudo mount -o remount,exec /home`

Change noexec bit to exec on /home partition.

## Preparing the Target on QNX\* RTOS

The following section applies only to target devices running QNX RTOS. After rebooting the system, complete the following steps on the target device once before running Intel SoC Watch collections:

1. Host: Find a host machine in the same subnet as the target system running QNX RTOS
2. Host: Set up an NFS share.

- a. Install the required packages:

```
sudo apt-get install nfs-kernel-server
sudo apt-get install rpcbind nfs-common
```

- b. Create a shared folder:

```
sudo mkdir /var/nfs
sudo chown user:user /var/nfs
```

- c. Add the target configuration to the "exports" file:

```
vim /etc/exports
/var/nfs xx.xx.xx.xx(rw,sync,no_subtree_check,insecure,all_squash,anonuid=1000,anongid=1000)
```

- d. Restart the nfs service:

```
sudo exportfs -ra
sudo /etc/init.d/nfs-kernel-server restart
```

- e. Check the shared folder configuration:

```
showmount -e <host-ip-address>
```

This should show `/var/nfs <target-ip-address>`.

3. Target: Map the NFS share to the root folder or a location of your choice.

```
fs-nfs3 xx.xx.xx.xx:/var/nfs /
```



- Host: Copy the Intel SoC Watch package to the NFS share to make it available on the target.

```
cp -r <extracted-package> /var/nfs
```

## Collect on Linux\* or Chrome\* 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.

### Set Up Collection on Chrome OS

To collect directly on the target, follow the instructions at <http://www.chromium.org/chromium-os/poking-around-your-chrome-os-device>. In general, place your device in Developer Mode, start a command prompt, and login as 'root' using the password you established when Intel SoC Watch was installed. Finally, mount the filesystem as described in the install directions.

To collect remotely on a target device running Google Chrome OS, use ssh to login to your target device as root using the password established in the install instructions. Finally, mount the filesystem as described in the install instructions.

### Collection on Linux or Chrome OS

- Load the device drivers. If the system is based on an Intel Atom® SoC, load the socperf3.ko driver. If the system is running the ACRN\* hypervisor, load the socwatchhv2\_0.ko driver. Then, load the socwatch2\_10.ko driver on all platforms. Note that the socperf3.ko driver must be loaded before the socwatch2\_10.ko driver.

Previous versions of the socwatch2\_x.ko driver (e.g. socwatch2\_0.ko) should work but new collector support and/or bug fixes may be missing in the older drivers. If an older socwatch2\_x.ko driver is used, some metrics may not be collected.

On systems running Chrome, if the socwatch\_chrome\_CUSTOM.tar.gz package was installed, issue the following commands:

```
insmod <path>/socwatch/socwatch_chrome_CUSTOM/driver/socperf3.ko
insmod <path>/socwatch/socwatch_chrome_CUSTOM/driver/socwatch2_10.ko
```

On systems running Linux, issue the following commands:

```
insmod <path>/socwatch/socwatch_chrome_linux_NDA_[version]/soc_perf_driver/src/socperf3.ko
insmod <path>/socwatch/socwatch_chrome_linux_NDA_[version]/socwatch_driver/socwatch2_10.ko
```

On systems running Linux as a virtual machine on top of a hypervisor, issue the following commands:

```
insmod <path>/socwatch/socwatch_linux_NDA_[version]/hypervisor_driver/src/socwatchhv2_0.ko
insmod <path>/socwatch/socwatch_linux_NDA_[version]/socwatch_driver/socwatch2_10.ko
```

- Navigate to the Intel SoC Watch directory.

```
cd <path>/socwatch/socwatch_[chrome_CUSTOM | chrome_linux_NDA_[version] |
linux_hypervisor_NDA_[version]]
```

- Setup the collection environment:

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

- Create a results directory:

```
mkdir results
```

- 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
```

6. View the summary results.

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

7. To view results in Intel VTune Amplifier 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. If the target system has an Intel Atom® processor, load the socperf driver. The socperf driver must be loaded before the Intel SoC Watch driver:

```
insmod <path_to_socperf_driver>/socperf3.ko
```

### NOTE

Make sure to use the latest version of the socperf driver.

3. Load the Intel SoC Watch driver:

```
insmod <path_to_socwatch_driver>/socwatch2_10.ko
```

4. Confirm the drivers are loaded:

```
lsmod
```

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

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

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

6. 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
```

7. Exit the adb shell:

```
exit
```

8. 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
```

## Collect on QNX RTOS

The QNX RTOS is a read-only file system. It is essential to run the Intel SoC Watch collection directly in a mounted NFS share.

### NOTE

Mount an NFS share following instructions in [Preparing the Target on QNX RTOS](#).

1. Change the directory to the Intel SoC Watch package folder on the NFS share

2. Set up the collection environment. This step assumes the / directory is where the NFS share is mounted.

```
cd /<soc-watch-package>
source ./setup_socwatch_env.sh
```

3. Collect data and generate 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
```

The results are automatically written on the NFS share and are available on the host. For example, /var/nfs on the host.

# 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 path and base filename (without extension) of an existing collection to generate additional reports. Use with the -r option to specify which types of reports.
-o	--output <filename>	Specify the base name for the output files from this collection. If this option is not specified, the files are written to the current working directory with base name SoCWatchOutput. Specifying console as the filename will cause the summary results to also write to stdout. If a name already exists, the previous results will be replaced.
-r	--result <result_type>	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: <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 Amplifier.</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	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.  Without this option, the tool collects data at the minimum required by the data source for best accuracy.  Data may be traced, sampled, or snapshot. On QNX RTOS, data may only be sampled or snapshot.

Abbreviation	Option Name	Description
		<ul style="list-style-type: none"> <li>Traced data is obtained at state transition points resulting in accurate summary and timeline results.</li> <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>--option &lt;opt-name=value&gt;</code>	Specify additional info for a metric. See <code>pch-ip-active</code> and <code>pch-ip-lat-limit</code> for valid option names and values.
	<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
all-approx-bw	Sampled	<p>Estimated memory bandwidth for multiple SoC agents (concurrently), from hardware signal accumulators. Precise bandwidth can be collected for one agent at a time using other bandwidth features. Only one bandwidth or DRAM self-refresh metric can be specified at a time.</p> <p>Hardware counter overflow can occur if -m is not used for signal-based metrics.</p> <p>SoC transactions are counted and multiplied by 64 to form the estimates, which will be equal to or higher than the actual bandwidths. The estimate is high if partial or 32-byte transfers actually occurred.</p> <p>On Intel platforms code named Broxton-M and Apollo Lake, this feature provides the following estimated bandwidths:</p> <ul style="list-style-type: none"> <li>• Module 0 and Module 1: the bandwidth from CPU Module 0 and Module 1 to the DDR</li> <li>• GFX: the bandwidth from the graphics component to the DDR</li> <li>• Display: the bandwidth from the display controller to the DDR</li> <li>• ISP: the bandwidth from the camera image processor to the DDR</li> <li>• VED: the bandwidth from the video encode and decode components to the DDR</li> <li>• IO: the bandwidth between the north and south clusters</li> </ul> <p>On 22nm Intel Atom® Processor-based SoC for Tablets and 2-in-1s (formerly code named Bay Trail), this feature provides the following estimated bandwidths:</p> <ul style="list-style-type: none"> <li>• Module 0 and Module 1: the bandwidth from CPU Module 0 and Module 1 to the DDR</li> <li>• GFX: the bandwidth from the graphics component to the DDR</li> <li>• Display: the bandwidth from the display controller to the DDR</li> <li>• ISP: the bandwidth from the camera image processor to the DDR</li> <li>• VED: the bandwidth from the video encode and decode components to the DDR</li> <li>• IO: the bandwidth between the north and south clusters</li> </ul> <p>On systems code named Cherry Trail, and Intel Atom Processor Z35XX (formerly code named Moorefield), this feature provides the following estimated bandwidths:</p> <ul style="list-style-type: none"> <li>• Module 0 and Module 1: the bandwidth from CPU Module 0 and Module 1 to the DDR</li> <li>• GFX: the bandwidth from the graphics component to the DDR</li> </ul>



Name	Collection Methods	Description
		<ul style="list-style-type: none"> <li>Display: the bandwidth from the display controller to the DDR</li> <li>ISP: the bandwidth from the camera image processor to the DDR</li> <li>IO: the bandwidth between the north and south clusters</li> </ul>
cnv-active	Snapshot	<p>WiFi Connectivity (CNV) component active residency, from hardware accumulator.</p> <p>This data can only be collected on Intel Atom Processor-based SoCs for systems code named Gemini Lake.</p>
cnv-ltr	Sampled	<p>WiFi Connectivity (CNV) component Latency Tolerance Reporting (LTR) summary, from hardware status data.</p> <p>This data can only be collected on Intel Atom Processor-based SoCs for systems code named Gemini Lake.</p>
core-temp	Sampled	IA core temperature statistics, from hardware status data.
core-volt	Sampled	<p>Calculate core voltage, from hardware status data.</p> <p>This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.</p>
cpu-ddr-mod0-bw cpu-ddr-mod1-bw	Sampled	<p>Precise CPU to DDR module 0 or 1 bandwidth, from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time.</p> <p>Hardware counter overflow can occur if -m is not used for signal-based metrics.</p> <p>Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.</p> <p>Also supported on Intel Atom Processor-based SoCs for systems code named Apollo Lake.</p> <p>Not available on QNX RTOS.</p>
cpu-gpu-concurrency	Snapshot	Concurrent active time of CPU and GPU, from hardware accumulators.
cpu-pkgc-cfg	Sampled	Platform settings that define CPU Package C-state entry behavior.
cpu-pkgc-dbg	Snapshot	Reasons blocking entry to lower-power C-states and wakeup from C-states for CPU Package, from hardware accumulators.
ddr-bw	Sampled	Total DDR memory bandwidth, from hardware accumulators or from signal accumulators. On platforms where signals are used, only one bandwidth or DRAM self-refresh metric can be specified at a time.

Name	Collection Methods	Description
		<p>The hardware accumulator data is always collected over time due to frequent overflow, so snapshot is not available.</p> <p>Hardware counter overflow can occur if -m is not used for signal-based metrics.</p>
deep-sstate	Sampled	Deep S0iX residencies, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.
die-temp	Sampled	Die temperatures, from PUNIT telemetry hardware status data. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.
disp-ddr-bw	Sampled	<p>Precise display controller to DDR memory bandwidth, from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time.</p> <p>Hardware counter overflow can occur if -m is not used for signal-based metrics.</p> <p>Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.</p> <p>Also supported on Intel Atom Processor-based SoCs for systems code named Apollo Lake.</p> <p>Not available on QNX RTOS.</p>
dram-bw	Snapshot	<p>Total DDR bandwidth with memory slice breakdown, from hardware accumulators. Available in place of ddr-bw on platforms where hardware signals not available.</p> <p>Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail, Broxton-M, Apollo Lake, and Gemini Lake.</p>
dram-pwr	Sampled	<p>Calculate DRAM power consumption, from PUNIT telemetry hardware accumulators.</p> <p>This data can be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Denverton, and on Intel systems code named Skylake-Xeon.</p>
dram-srr	Sampled	<p>DRAM residency in self-refresh mode, from hardware signals. Only one bandwidth or DRAM self-refresh metric can be specified at a time.</p> <p>Hardware counter overflow can occur if -m is not used for signal-based metrics.</p> <p>Do not specify a polling interval greater than 14 seconds when measuring DRAM self-refresh.</p> <p>Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.</p>

Name	Collection Methods	Description
		Also supported on Intel Atom Processor-based SoCs for systems code named Apollo Lake. Not available on QNX RTOS.
edp-throt-cnt	Sampled	Throttling counts due to EDP (electrical design power) limits, from PUNIT telemetry hardware status data. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Gemini Lake.
edp-throt-res	Sampled	Throttling residencies due to EDP (electrical design power) limits, from PUNIT telemetry hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Gemini Lake.
fpga-bw-read	Sampled	PCIe Read bandwidth by the FPGA.
fpga-bw-write	Sampled	PCIe Write bandwidth by the FPGA.
fpga-pwr	Sampled	Power consumed by the FPGA.
fpga-temp	Sampled	Temperature of the FPGA.
gfx-ddr-bw	Sampled	<p>Graphics component to DDR bandwidth, from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time.</p> <hr/> <p><b>NOTE</b> Some bandwidth measurements include ReadPartial or WritePartial traffic. The payload (how many bytes were transferred) of a partial transaction can range from 1 to 64 bytes. Therefore, the exact bandwidth cannot be accurately measured. In these cases, the Intel SoC Watch bandwidth results will provide a Total Bandwidth Range. The Total Bandwidth Range describes the minimum and maximum bandwidth that was measured. The actual bandwidth falls within the Total Bandwidth Range.</p> <hr/> <p>Hardware counter overflow can occur if -m is not used for signal-based metrics. Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M. Also supported on Intel Atom Processor-based SoCs for systems code named Apollo Lake. Not available on QNX RTOS.</p>
igfx-throt-rsn	Sampled	Reasons for throttling the integrated GPU frequency, from hardware status data.
gt-pwr	Sampled	Calculate GT power consumption, from PUNIT telemetry hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.

Name	Collection Methods	Description
gt-temp	Sampled	GT graphics temperature, from hardware status data. This data can only be collected on Intel Atom® Processor-based SoCs for systems code named Apollo Lake and Broxton.
hw-cpu-cstate	Snapshot Snapshot , Trace Android OS, Chrome OS, and Linux OS: Snapshot, Traced QNX RTOS: Snapshot, Sampled	CPU C-state (sleep) residencies for package/module/core, from hardware accumulators, and summary of wakeups that cause IA cores to exit a C-state from trace data. Wakeups are only collected if the <code>-m</code> or <code>--max-detail</code> switch is specified. Wakeups are not available on QNX.
hw-cpu-hwp	Sampled	HWP capabilities, from hardware status data.
hw-cpu-pstate	Sampled	CPU P-state operating frequency residencies, from trace data.
hw-igfx-cstate	Snapshot	Integrated graphics processor C-state residency (RC6), from hardware accumulators. Always sampled due to short overflow time period.
hw-igfx-pstate	Sampled	Integrated graphics processor P-state operating frequency residencies, from hardware status data.
ia-pwr	Sampled	Calculate IA power consumption, from PUNIT telemetry hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.
ia-temp	Sampled	IA core temperature, from hardware status data. This data can only be collected on Intel Atom® Processor-based SoCs for systems code named Broxton-M, Apollo Lake and Gemini Lake.
ia-throt-rsn	Sampled	Reasons for throttling the CPU frequency, from hardware status data.
io-bw	Sampled	IO bandwidth between the North Cluster (NC) and South Cluster (SC), from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time.  Hardware counter overflow can occur if <code>-m</code> is not used for signal-based metrics.  Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.  Also supported on Intel Atom Processor-based SoCs for systems code named Apollo Lake.  Not available on QNX RTOS.
isp-ddr-bw	Sampled	ISP (camera image processor) to DDR bandwidth, from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time.

Name	Collection Methods	Description
		<p>Hardware counter overflow can occur if -m is not used for signal-based metrics.</p> <p>Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.</p> <p>Also supported on Intel Atom Processor-based SoCs for systems code named Apollo Lake.</p> <p>Not available on QNX RTOS.</p>
isp-pwr	Sampled	Calculate ISP power consumption, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.
isp-temp	Sampled	ISP (camera image processor) temperature, from hardware status data. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Broxton and Apollo Lake.
iunit-temp	Sampled	IUNIT temperature, from hardware status data. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.
lpss-ltr	Sampled	Low Power Subsystem (LPSS) Latency tolerance reporting (LTR) for hardware and software mechanisms, from hardware status data.
mod-temp	Sampled	(NDA) Module temperature, from hardware status data. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.
nc-dstate	Sampled	<p>North Complex (NC) component D0ix approximated state residency, from hardware status data.</p> <p>This data can only be collected on Intel Atom Processor-based SoCs for systems code named Cherry Trail.</p>
netip-bw	Sampled	<p>Networking IP block to DDR bandwidth, from hardware signal accumulators. Only transfers of 32 or 64 bytes are measured. Use both the <code>netip-bw</code> and <code>netip-partials-bw</code> features to measure the full networking IP block to DDR bandwidth. Only one bandwidth or DRAM self-refresh metric can be specified at a time.</p> <p>Hardware counter overflow can occur if -m is not used for signal-based metrics.</p>
netip-partials-bw	Sampled	Networking IP block to DDR bandwidth, from hardware signal accumulators. Only partial transfers are measured (partial transfers are non-32byte or non-64byte transactions). Use both the <code>netip-bw</code> and <code>netip-partials-bw</code> features

Name	Collection Methods	Description
		to measure the full networking IP block to DDR bandwidth. Only one bandwidth or DRAM self-refresh metric can be specified at a time.  Hardware counter overflow can occur if -m is not used for signal-based metrics.
other-throt-cnt	Sampled	Throttling counts due to causes other than power, thermal, or EDP (electrical design power) limits, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Gemini Lake.
other-throt-res	Sampled	Throttling residencies due to causes other than power, thermal, or EDP (electrical design power) limits, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Gemini Lake.
panel-srr	Sampled	Panel self-refresh residency report, from hardware status data.
pch-ip-active	Snapshot	<p>PCH active residencies for each IP, from hardware accumulators. To avoid interference in attaining deeper sleep states on older PCH, the PCH activity data is split into multiple groups referred to as levels (lvl) and data is collected at a much lower rate. By default, lvl1 is reported and the collection interval is 5 seconds. Additional levels are shown below, as is the option to change the collection interval.</p> <p>To specify a level, include <code>--option pch-lvl=&lt;lvl, lvl, ...&gt;</code>. Listing multiple levels is not allowed for older PCH due to impact of overhead. All levels are not supported for all PCH and will be ignored if specified. IP data included in each level can vary for each PCH. Valid values for lvl are:</p> <ul style="list-style-type: none"> <li>1: Includes PMC, PLL activity for OPI, SATA, Audio, USB, MIPI, and CNVi, and more</li> <li>2: Audio high speed fabric, SCS, Thermal Sensor, and ISH, and more</li> <li>3a: ModPhy lanes 0-3</li> <li>3b: ModPhy lanes 4-7</li> <li>3c: ModPhy lanes 8-11</li> <li>3d: ModPhy lanes 12-15</li> <li>4: Power-gated domains, ungated SRAMS, camera, and more</li> <li>5: LPSS, CNVi components, CRO, XTAL, CLKOUT, and more</li> </ul> <p>To change the frequency at which this data is collected, include <code>--option pch-int=&lt;milliseconds&gt;</code>. Use of intervals below the</p>

Name	Collection Methods	Description
		default 5 seconds for PCH active metrics has higher impact on overhead and may reduce time in low power states. Use the <code>--interval</code> option for all other metrics.
<code>pch-ip-active-all</code>	Snapshot	All levels of PCH active residencies for each IP, from hardware accumulators. Not available on platforms with high-overhead mechanism for collecting this data, use the <code>pch-ip-active</code> feature with its <code>pch-lvl</code> option to specify one group at a time.
<code>pch-ip-lat-limit</code>	Snapshot	PCH IP latency limit residencies, from hardware accumulators. Exactly three IPs can be monitored at the same time. By default SPA, SPB, and SPC will be monitored.  To change which IPs to monitor, include <code>--option lat-limit-ip=&lt;IP1, IP2, IP3&gt;</code> , where each IP is one of the following: SPA, SPB, SPC, SATA, GBE, XHCI, ME, EVA, HD-AUDIO, ESPI, LPSS, CAM, SCC, ISH. The exact list of IPs can vary for different PCH. Use the <code>--help</code> option to see which IP are available on your target platform.
<code>pch-ip-status</code>	Sampled	PCH IP blocks power gated statistics, from hardware status data.
<code>pch-platform-ltr</code>	Sampled	PCH IP Latency Tolerance Requirement (LTR) for snoop and no snoop, from hardware status data.
<code>pch-slps0</code>	Snapshot	PCH SLP_S0 residency, from hardware accumulator.
<code>pch-slps0-dbg</code>	Sampled	Blocking reasons for SLP_S0, from hardware status data.
<code>pcie-lpm</code>	Sampled	PCIe Link Power Management (LPM) Lx state residencies, from hardware status data. Includes root port and end point device capabilities. L1 sub-states are included if supported by the device (Gen 1, Gen 2, Gen 3).
<code>pcie-ltr</code>	Sampled	PCIe Latency Tolerance Reporting (LTR) for snoop and no-snoop requests, from hardware status. Includes root port and end point device capabilities.
<code>pkg-pwr</code>	Snapshot	Calculate the entire SoC/Package power consumption, from hardware accumulator.
<code>pkg-temp</code>	Sampled	Package temperature, from hardware status data. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Broxton and Apollo Lake.
<code>pmc-ip-pg-res</code>	Snapshot	Power gated residency for each PMC component, from hardware accumulators.

Name	Collection Methods	Description
pmc-ip-status	Sampled	Approximated time in power-gated, D0i3, and D3 states for each PMC component, from hardware status data.
pmic-imon	Sampled	Average current on multiple PMIC voltage rails, from hardware status data.
pmic-pwr	Sampled	Calculate PMIC rails power values, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.
pmic-temp	Sampled	MSIC/PMIC temperature, from sysfs reads.
pwr-throt-cnt	Sampled	Throttling counts due to power limits, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Gemini Lake.
pwr-throt-res	Snapshot	Throttling residencies due to power limits, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Gemini Lake.
ring-throt-rsn	Sampled	Reasons for throttling the ring clock frequency, from hardware status data.
s0i3-sstate	Snapshot	S0i3 system state residencies, from hardware accumulators.  This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Gemini Lake.
s0i3-sstate-dbg	Snapshot	Blocking reasons for entry to system state S0i3 (wake up reasons for exit from S0i3 are included on some platforms), from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake
s0ix-sstate	Sampled	S0ix system state residencies.  This data can only be collected on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Anniedale .)
s0ix-subs-res	Snapshot	S0ix system sub-state residencies, from hardware accumulators.  This data can only be collected on Intel platforms code named Tiger Lake.
s0ix-subs-status	Sampled	S0ix system sub-state status of low power mode requirements, from hardware status data.  This data can only be collected on Intel platforms code named Tiger Lake.



Name	Collection Methods	Description
sa-temp	Sampled	System Agent temperature, from hardware status data. This data can only be collected on Intel Atom® Processor-based SoCs for systems code named, Broxton-M, Apollo Lake and Gemini Lake.
sata-lpm	Sampled	SATA Link Power Management (LPM) states residencies, from hardware status data. Includes device capabilities.
sc-dstate	Snapshot or Sampled (depends on target system)	South Complex (SC) component D0ix state residencies. If sampled, the residency is only an approximation.  This data can only be collected on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Anniedale .
shallow-sstate	Sampled	Measure shallow S0ix residencies, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.
skin-temp	Sampled	Skin temperature data, from sysfs reads.
soc-temp	Sampled	SoC temperature data, from hardware status data or PMC telemetry. This data can only be collected on Intel Atom® Processor-based SoCs for systems code named Cherry Trail, Broxton-M, Apollo Lake and Gemini Lake.
tcss-state	Snapshot	TypeC subsystem state (TCx) residency and entrance count, from hardware accumulators.
tcss-cfg-status	Sampled	TypeC subsystem (IOM, TBT) configuration status, from hardware status data.
temp-throt-cnt	Sampled	Throttling counts due to thermal limits, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Gemini Lake.
temp-throt-res	Sampled	Throttling residencies due to thermal limits, from hardware accumulators. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Gemini Lake.
vm-switch	Trace	Virtual machine context switch activity, from tracepoints in the virtual machine hypervisor. Switches between the hypervisor, Android virtual machine (VM), service OS VM (if present), Modem VM (if present), and Secure VM (if present) are tracked. The residency of the hypervisor and the Guest Oses are provided. VM Exit causes are also described in the results.
wakelock	Trace	User and kernel wakelock data, from kernel tracepoints and aplog. Android only.

Name	Collection Methods	Description
xhci-lpm	Sampled	XHCI Link Power Management state (Ux) residencies, from hardware status data. Includes capabilities of the devices directly attached to an XHCI controller operating in USB 2.0 or 3.0 modes.  Chrome or Linux only.

## 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
cnv	CNVi (WiFi) metrics such as active and LTR.
cpu	cpu-hw
cpu-hw	Most CPU metrics obtained from hardware data sources.
device	All I/O device metrics.
display	panel-srr
gfx	All graphics metrics from hardware and OS. gfx-hw
gfx-hw	Most GPU metrics obtained from hardware data sources.
hw-gfx-cstate	hw-igfx-cstate
hw-gfx-pstate	hw-igfx-pstate
lpss	Low power subsystem I/O metrics.
pcie	PCIe metrics such as link state residencies and latency tolerance requirements.
pch	Most PCH related metrics.
pch-all	All PCH related metrics (all PCH active groups).
power	Power/energy metrics.
sata	SATA related metrics.
sstate	System Sx and S0ix state metrics.
sys	Broad spectrum of metrics commonly used to get general information about platform power behavior.
tcss	TypeC sub-system metrics.

<b>Name</b>	<b>Description</b>
temp	Most temperature metrics.
throt	Frequency throttling reason metrics.
xhci	XHCI related metrics.

## ***Viewing Intel SoC Watch Results with Intel® VTune™ Amplifier***



You can analyze Intel SoC Watch data graphically using the Intel® VTune™ Amplifier GUI. Intel® VTune™ Amplifier 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 [Analyze Energy Usage](#) section of the Intel® VTune™ Amplifier Help.