Intel® SoC Watch for Linux* Release Notes

November 2020

Version 2020.5

Intel Corporation

www.intel.com

Legal Information

Contents

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'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.

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.

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 *integrated* to hw-igfx-cstate and hw-igfx-pstate report titles. Fix for hw-cpu-cstate reporting of core concurrency.

Customer Support

For technical support, including answers to questions not addressed in this product, Intel® oneAPI IoT Toolkit forum (https://community.intel.com/t5/Intel-oneAPI-IoT-Toolkit/bd-p/oneapi-iot-toolkit).

1 Introduction

Intel® SoC Watch is a data collector for power-related data that can help identify issues on a platform that prevent entry to power-saving states. Captured metrics include:

- System sleep states
- CPU and GPU sleep states
- Processor frequencies
- Temperature data
- Device sleep states

You can correlate the collected data and visualize over time using Intel®VTune Profiler.

This document provides system requirements, installation instructions, issues and limitations, and legal information.

To learn more about this product, see:

- New features listed in the New in This Release section below, or in the help.
- Reference documentation listed in the Related Documentation section below
- Installation instructions can be found in the Installation Notes section below.
- For a detailed quick start guide to running the tool, see the *Intel SoC Watch User's Guide* in your installed documentation.

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

2 New in This Release

The 2020.5 release (driver v2.13.1) contains these changes:

- Feature -f hw-cpu-pstate now summarizes and reports CPU P-states per thread (logical processor) rather than per core. The per core summary gave inaccurate results on all new platforms that are running with Hyper-Threading, starting with Intel platform code named Ice Lake. Previously, SoC Watch displayed a warning about inaccurate P-state data and suggested use of –polling option as a work around on those platforms. Additional notes:
  - This change resolves occasional hardware counter correctness issue which resulted in SoC Watch displaying a warning about invalid P-state data because different counters are now used.
  - Added hidden option -f hw-cpu-pstate-per-core to get the old report for use on platforms prior to Ice Lake, where P-state collection per Core is still valid. This feature will report inaccurate results if used on Ice Lake or newer platforms unless the --polling option is included. Note: Features hw-cpu-pstate and hw-cpu-pstate-per-core are mutually exclusive. Attempting to collect both in the same run will result in an error.
- Changed table names for features -f hw-igfx-cstate and -f hw-igfx-pstate to include "Integrated". Previously, these output tables did not identify the results as Integrated Graphics vs. Discrete Graphics.
- Support for the following older platforms has been removed in this release: Intel platforms code named Anniedale, Valleyview, Cherry Trail, Haswell, and Broadwell.

3 System Requirements

Supported Architectures

Intel SoC Watch supports these Intel microarchitecture or platform code names:

- Denverton
- Apollo Lake
- Gemini Lake
- Skylake
- Kaby Lake
- Coffee Lake
- Whiskey Lake
- Amber Lake
- Comet Lake
- Ice Lake
- Tiger Lake
- Skylake-Xeon
- Cascade Lake-Xeon

Intel SoC Watch supports these Intel discrete graphics code names:

- DG1

Dependencies

Intel SoC Watch depends on specific OS configurations and hardware capabilities. If these are not present on the target system, Intel SoC Watch may fail to work properly.

- Linux Kernel version needs to be 2.6.32 or later.
- GNU C Library version must be GLIBC_2.17 or later.
- KERNEL_CONFIG_TRACEPOINTS must be enabled.
- Kernel should be compiled with "CONFIG_MODULES" enabled.
- P States
  - Kernel config CONFIG_X86_SFI_CPUFREQ or CONFIG_X86_ACPI_CPUFREQ must be enabled (i.e. set to 'y' or 'm').
  - One of these pstate drivers must be utilized: sfi-cpufreq, acpi-cpufreq, or intel_pstate. To determine which driver is loaded, check the sysfs /sys/devices/system/cpu/cpu0/cpufreq/scaling_driver file.
  - If one of these pstate drivers is not loaded, the kernel needs to be reconfigured and recompiled.
- C States
  - Kernel config CONFIG_TIMER_STATS must be enabled.
  - Kernel config CONFIG_INTEL_IDLE must be enabled and the intel_idle kernel module has to support the core of the target platform.
  - To determine if the intel_idle kernel module is loaded, check the sysfs /sys/devices/system/cpu/cpuidle/current_driver file. It must equal intel_idle. If it equals acpi_idle, only C0 and C1 will be used by the core.

4 Where to Find the Release

Intel® SoC Watch is available in Intel® oneAPI Base Toolkit as a part of Intel® VTune™ Profiler (https://software.intel.com/content/www/us/en/develop/tools/oneapi/base-toolkit).

5 Installation Notes

Intel SoC Watch for Linux* OS is installed as part of the Intel® VTune™ Profiler, which is included in the Intel® oneAPI Base Toolkit. Intel SoC Watch is used by Intel® VTune™ Profiler for collecting certain analysis, but it is also a standalone tool. Following are instructions for setting up Intel SoC Watch for standalone use.

Extracting the Intel SoC Watch Package

Intel SoC Watch must be installed on the target system (the system to be analyzed). To extract and install the package, the user should have sudo (root) privilege.

When Intel® oneAPI Base Toolkit is used to install Intel® VTune™ Profiler and it is installed directly on the target system:

Intel SoC Watch will already be extracted and is located in `/opt/intel/oneapi/vtune/<oneAPI-release-version>/socwatch`.

The Intel SoC Watch release notes and user's guides are located in `/opt/intel/oneapi/vtune/<oneAPI-release-version>/documentation/en/socwatch`.

When Intel® oneAPI Base Toolkit is used to install Intel® VTune™ Profiler and it is installed on a host system:

Intel SoC Watch will need to be copied to the target system and extracted. The install package is located on the host system at `/opt/intel/oneapi/vtune/<oneAPI-release-version>/target/linux/vtune_profiler_target_x86_64.tgz`. Copy `vtune_profiler_target_x86_64.tgz` to the target system and extract it into directory `/opt/intel` using command `tar zxvf vtune_profiler_target_x86_64.tgz -C /opt/intel/`.

The release notes and user's guide are not included in this package. They are located on the host system in `/opt/intel/oneapi/vtune/<oneAPI-release-version>/documentation/en/socwatch`.

You must build the kernel modules and install the SoC Watch driver to enable collection of all metrics. See instructions below.

Intel SoC Watch Prerequisites for Yocto and Wind River* Linux

The Intel SoC Watch binary is a C++ program that requires libstdc++ to be present on the target system in order to function. Although most Linux distributions provide this by default, there are some minimal distributions of Yocto and Wind River Linux that may not.

To check if your OS image contains libstdc++, run this command:

```
find / -name "libstdc++*"
```

If the library is present, you should see an output similar to:

```
/usr/lib64/libstdc++.so.6
```

If you see no output, the required library is missing. To fix this, do one of these steps:

- Rebuild Yocto or Wind River Linux with an option to include libstdc++.so file in the image.
- Provide the library file directly to Intel SoC Watch.

Rebuild OS image to include libstdc++.so file

1. Open the projectDir/local.conf file in a text editor.
2. Add the library with the IMAGE_INSTALL_append option and save the file:

```
IMAGE_INSTALL_append = " libstdc++"
```

NOTE Make sure that you include the leading space as it will be concatenated with any libraries previously added in a list.

3. Rebuild the platform project by running the following command from the projectDir:

```
make
```

Once the build completes, the library will be part of the project image rootfs. Proceed with reflashing the OS image to the target system and subsequently installing Intel SoC Watch.

Provide libstdc++.so library file directly to Intel SoC Watch

If you are unable to rebuild the OS image, but are able to obtain a valid libstdc++.so file (you can copy it from another system or build it yourself):

1. Unpack the SoC Watch package.
2. Copy the file into the /libs subfolder where it will be picked up at runtime.

libgcc_s.so Library

Some minimal Yocto distributions may also lack another library: libgcc_s.so. In this case, follow either of the aforementioned solutions.

Build the OS and include the image:

Use this command when building the OS.

```
IMAGE_INSTALL_append = " libgcc_s"
```

Copy file into Intel SoC Watch package:

Copy the library libgcc_s.so file into the /libs folder of the Intel SoC Watch package.

Build the Kernel Modules

If the Intel SoC Watch kernel modules (i.e. device drivers) are not present in the OS image of the target system, you will need to build and possibly sign them. Building and signing device drivers requires access to the kernel build directory for the OS image running on your target device. A kernel build directory is generated while building the OS image of the target system.

When building the kernel modules, do not open (or unzip) the Intel SoC Watch package (i.e. tar.gz file) on a Windows* based system then copy to a Linux system. The package must be extracted on the Linux build system using the unzip command to make sure the build scripts and make files are unmodified.

If a kernel is built with the CONFIG_MODULE_SIG kernel config enabled, any device driver loaded into that kernel must be signed with the same keys used to build the kernel. In general, drivers built for Linux targets do not need to be signed and the following description assumes the drivers do not need to be signed. But, if an end user tries to load an unsigned driver into a kernel that requires signed drivers, the `insmod` command will fail with the error "Required key" not available. If a signed driver is loaded into a kernel that does not require signed drivers, the load will succeed.

Building Linux* Kernel Modules

Linux kernel modules may only be built after the Intel SoC Watch package and kernel headers are copied to and installed on the target. See the section Intel SoC Watch for Linux Installation below for instructions on how to build the kernel modules for a target device running Linux.

Install Intel SoC Watch

Host: laptop, desktop, or server used to communicate with target device.

Target: device to be analyzed with Intel SoC Watch.

Intel SoC Watch for Linux* Installation

If Intel SoC Watch was previously installed on the target, delete the `socwatch_linux_*` directory before installing a new version. Then, perform the following steps on the target device.

1. Login to the target as root:

```
ssh root@<your_target_IP>
```

2. Identify the <install-dir> where SoC Watch was installed as indicated below. Note that the SoC Watch driver build must be executed on the OS kernel that matches the target and the driver install must be executed on the target system. If Intel SoC Watch was installed on a host system, then it must be copied to the target before proceeding. (See Extracting the Intel SoC Watch Package)
   a. *Host system:* `/opt/intel/oneapi/vtune/<oneAPI-release-version>/`
   b. *Target system:* `/opt/intel/vtune_profiler_target`
3. Navigate to the Intel SoC Watch directory:

```
cd <install-dir>/sepdk/src
```

4. Use the following commands to build the SoC Watch driver:
   a. To build the driver.

```
./build-drivers -ni                              (switch "ni" refers to "No User Interface")
```

   b. Reload the module to load the SoC Watch driver.

```
./insmod-sep -r
```

   c. To verify if SoC Watch driver is loaded, use the below command to query.

```
./insmod-sep q
```

NOTE
You may see a compile-time error related to "freqs->policy->cpu" on certain Linux distributions (Chrome, Redhat) that have backported a change from kernel 5.2 into 4.19 which breaks compatibility. As we have no clean way to know whether this has been done, it requires a manual change to accommodate. The changes below must be done to avoid the compile-time error when running `build_drivers.sh`

Comment out the lines below in "`socwatch_driver/./src/sw_trace_notifier_provider.c`" so that the newer version (freqs->policy->cpu) is used instead.

```
//#if KERNEL_VERSION(5, 2, 0) > LINUX_VERSION_CODE

//      int cpu = freqs->cpu;

//#else  /* KERNEL_VERSION(5, 2, 0) <= LINUX_VERSION_CODE */

        int cpu = freqs->policy->cpu;

//#endif /* KERNEL_VERSION(5, 2, 0) > LINUX_VERSION_CODE */
```

5. Use the below command to configure SoC Watch collection environment . To run on
   a. *Host system:* `source <install-dir>/vtune-vars.sh`
   b. *Target system:* `source <install-dir>/sep_vars.sh`
   c. To set the environment variable, use the following command (The <install-dir> should be based on the host/target system.)

```
export PATH=<install-dir>/socwatch/x64:$PATH
```

NOTE Sourcing of `vtune-vars.sh` or `sep_vars.sh` is required if `/opt/intel/oneapi/setvarsh.sh` is not already sourced as a part of Intel® oneAPI Base Toolkit installation.

Prerequisites for FPGA metric collection using Intel SoC Watch

Intel SoC Watch leverages Open Programmable Acceleration Engine (OPAE) drivers to collect data on supported FPGA platforms. OPAE drivers must be loaded on the system containing one of the supported FPGA architectures (see System Requirements) prior to running Intel SoC Watch collections. Source code for OPAE drivers may be downloaded from the OPAE GIT repository located at the following link: https://github.com/OPAE/opae-sdk/releases

The OPAE installation guide provides more information on the installation requirements for OPAE drivers. The guide may be downloaded from the following location: https://opae.github.io/latest/docs/install_guide/installation_guide.html.

Follow these instructions to build the OPAE drivers for your system:

1. Download the kernel headers to your system.
2. Download and extract the file starting with `opae-intel-fpga-driver*.tar.gz` to your system.
3. Go to the extracted directory, locate the file named `Makefile`, and run the `make` command in that directory.

This process will generate several files with a `.ko` extension. These are the OPAE drivers that need to be loaded to enable Intel SoC Watch collections. Load these driver files using the following commands:

```
sudo insmod fpga-mgr-mod.ko
sudo insmod intel-fpga-pci.ko
sudo insmod intel-fpga-fme.ko
sudo insmod intel-fpga-afu.ko
```

Key Files

The following table describes the key files.

File	Description
`build_drivers.sh`	The build script used to build all of the device drivers utilized by Intel SoC Watch.
`socwatch2_x.ko`	The Intel SoC Watch kernel module used to collect both hardware and kernel data at runtime.
`setup_socwatch_env.sh`	The script used to setup the Intel SoC Watch runtime environment.
`socwatch`	The Intel SoC Watch executable built as a native application. Use this file to collect data and generate additional results from a raw SW2 file.
`SOCWatchConfig.txt`	The Intel SoC Watch configuration file. The configuration file is read by Intel SoC Watch immediately before each collection. It contains hardware addresses utilized by the device driver during the collection.
`EULA.txt`	End User License Agreement file.
`third-party-programs.txt`	List of third party programs included in the package.

File	Description
`plugins/libSWCore.so`	A library providing Intel SoC Watch functionality.

Remove the Intel SoC Watch Drivers

Remove Intel SoC Watch drivers using the `rmmod` command. For example: `rmmod socwatch2_13`.

6 Fixed Issues

The 2020.5 release has a fix for these issues.

- Improved Core Concurrency report for feature -f hw-cpu-cstate (seen when -m option is specified). Previously, active time reported in the concurrency report was too high. It was inconsistent with the C-state residency report that showed time in lower-power sleep states, when cores would not have been active.

7 Known Issues

Bandwidth

- Memory bandwidth metric (-f ddr-bw) is not currently supported on Intel platforms code named Tiger Lake.
- The presence of EDRAM on a system may not be detected by Intel SoC Watch. This is known to occur when the accelerator card VCA2, which contains EDRAM, is present.
- Total DDR bandwidth does not include EDRAM. On systems using EDRAM, the `ddr-bw` feature report may have a discrepancy between the total data reads and writes and the total component requests. The Data Reads+Data Writes will be significantly higher than the total IA+GT+IO requests, because the EDRAM requests are not included.

C-States / P-States

- When collecting a trace of residency data from hardware counters (i.e., using -m), the summarized residency data could be 2-3% inaccurate due to error propagation in the accumulation of each sample's calculated residency. Collecting without -m results in greater accuracy because only a single sample is taken. However, long collection duration could result in a counter rollover, and that will not be detected without the use of -m.
- The hardware CPU P-state data may be missing for some cores when using feature `-f hw-cpu-pstate` on Intel platforms code named Skylake, Kaby Lake, Whiskey Lake, and Amber Lake. The issue is caused by unexpected behavior of the hardware counters. The tool ignores these bad samples which results in the missing data. This issue is resolved in Intel SoC Watch v2020.5 where hw-cpu-pstate was modified to use different counters and report P-states per thread rather than per core. The problem can occur if hidden metric hw-cpu-pstate-per-core is used.
- On Intel platforms code named Broxton and Apollo Lake, the `cpu-cstate` metric results do not contain module C-state information.
- On Intel Atom® platforms, if all cores in a module request C6FS but actual sleep time is short, the Auto-Demotion logic of the hardware resolves the module state to module C0. Consequently, you may find module C0 to be greater than the sum of core C0 and C1 on all the cores in a module. On the same lines (auto demotion at the package level), the package C0 may be greater than module C0 residencies of the two modules.
- During the transition time from core C1/C1e/C6 to core C0, a core may run in LFM which will be properly measured by Intel SoC Watch. Therefore, results that include a large number of C1/C1e/C6 residencies may show a lower PState than expected.

Miscellaneous

- On ChromeOS, some SoCWatch collections may delay their start until after a waking up from S0iX sleep under certain conditions. There are two known alternative solutions: 1) Unload the socwatch.ko kernel module, allowing socwatch to run in "driverless" mode. 2) Avoid the use of wrapper scripts to put the system into S0ix and instead use SoCWatch's built-in capability for this (the "-z" feature switch). This will force the system to sleep for the duration of the "-t" switch, and relevant outputs will reflect the time spent in sleep.
- When running on Linux Yocto OS, SoC Watch will simply return without executing unless you create a symbolic link for /lib64 pointing to /lib.
- If collecting a large number of metrics and requesting multiple types of results files to be generated on the same command line, SoC Watch may report the following, *Warning: Could not post process metric data: Too many open file handles. Results may be incomplete for some metrics. Try post processing results with only one -r option at a time. If the problem persists; try collecting fewer metrics.* There will

be some missing reports in the results files that are generated if this occurs. The work around is to specify only one -r file type at collection time or collect fewer metrics. After collection, use -i option with each of the remaining file types to generate the additional result files.

- Feature -f dram-pwr is not supported by all versions of the server Intel platforms code named Skylake-Xeon, Cascade Lake-Xeon, and Denverton). The report contains all zero values in this case.
- Metrics report Unknown 0 when `-m` is not used and hibernation occurs. Metrics with a snapshot default collection mode, such as CPU C-state, will show the Unknown state with 0 time and the remaining states will not sum to the total collection duration if the system entered hibernation during the collection and the `-m` option was not specified. The snapshot metrics are only collected at the start and end of a collection by default, but finding hibernation time requires samples taken throughout the collection. Including `-m` will cause continuous sampling to occur for all metrics. When hibernation occurs, a message reporting time spent in hibernation appears at the beginning of the summary report. The Unknown state is then included for all appropriate metrics and the time in hibernation is included in that state. Refer to the *Intel SoC Watch User's Guide* "Options Quick Reference" section to learn which metrics have a snapshot collection mode by default.
- Intel SoC Watch reads PMIC and Skin Temperatures from the system's sysfs. Rarely, a sysfs read may not return before a subsequent sysfs read occurs. When this occurs, specific sample results may be missing in the timed trace CSV and raw text files.
- Permission issues with SELinux will cause Intel SoC Watch collection to fail. Some distributions enable SELinux by default. If you have the following file your system may have SELinux enabled:

```
/selinux/enforce
```

  If that file exists, you can disable by issuing:

```
echo 0 > /selinux/enforce
```

- Syntax errors in the command line may not report a visible error message. If a collection did not run and you are not seeing any error message, add option `-d 2` to your command line to get more information.

Intel® VTune™ Profiler Visualization

- Importing a collection that includes feature –f dgfx-pwr to Intel® VTune™ Profiler may fail with "Database interface, Precompute error". This issue was first seen with Intel® VTune™ Profiler v2020 Update 3.
- When using Intel® VTune™ Profiler to view ddr-bw data collected by SoC Watch, the Intel VTune Profiler summary report may not match the SoC Watch summary report. If this occurs, the SoC Watch summary results are correct. This issue has been seen on platforms prior to Intel platform code named Tiger Lake.
- The Intel VTune Profiler System Summary does not report the rated frequency for the CPU when viewing results from data collected by SoC Watch, such as Throttling Analysis. Instead, it reports 1GHz which is the clock frequency used in the calculations for processing the SoC Watch data.
- Intel VTune Amplifier 2017 for Systems Update 1 or later is required for visualizing and analyzing Intel SoC Watch v2.10.0 and newer PWR files. We recommend using the latest version of Intel VTune Profiler.
- If the bandwidth is 0 Mb throughout the collection for a particular bandwidth type, Intel VTune Profiler will not show a timeline entry for it. The timeline is shown only if there is at least one non-zero value.
- In some cases, the summary CSV results produced by Intel SoC Watch can vary from the summary results shown by Intel VTune Profiler even though they represent the same collection. For example, the summary CSV file may report a specific cpu-pstate residency of 50.78% and Intel VTune Profiler may report the same cpu-pstate residency as 50.8%.
- Intel VTune Profiler currently does not support bandwidth ranges used for ReadPartial and WritePartial. In order to keep the visualization consistent with Intel SoC Watch v1.x, Intel VTune Profiler uses the upper bound of the range to visualize the bandwidth.
- In order to visualize graphics C-states that are reported as Render and Media, the table headers in the trace file (generated with option -r int), must be manually modified, adding *Render* and *Media* to the appropriate C0, C1, and C6 column headers.

8 Related Documentation

The release contains these documents:

- Intel® SoC Watch for Android* OS and Linux* OS User's Guide
- Energy Analysis help (https://software.intel.com/content/www/us/en/develop/documentation/energy-analysis-user-guide)