Intel® Energy Checker Frequently Asked Questions

Q1 - What is the Intel® Energy Checker API?

A1 - The Intel® Energy Checker (Intel® EC) API provides the functions required for exporting and importing counters from an application. A counter stores the number of times a particular event or process has occurred, much like the way an odometer records the distance a car has traveled. Other applications can read these counters and take actions based on current counter values or trends derived from reading those counters over time. The core API consists of five functions to open, re-open, read, write, and close a counter.

Q2 - Why is the API so simple?

A2 - It is simple by design! We defined the API and the associated methodologies to make it as simple as possible for ISVs to instrument their code. We constantly pushed back on the addition of nice-to-have features to stay simple.

Q3 - What is the Intel® Energy Checker SDK?

A3 - This Software Development Kit (SDK) enables Independent Software Vendors (ISV) to facilitate the analysis of an application's energy efficiency. The SDK provides a simple API and the tools required to define, measure and share energy efficiency data. ISVs can instrument their applications' source code to export and import counters in a standard way. Although the initial intent of this SDK is to facilitate energy efficiency analysis and optimizations, it can be used to expose any counter meaningful to the ISV and its customers.

Q4 - How do I download the Intel® Energy Checker SDK?

A4 - Visit the end user agreement and download page for the Intel® Energy Checker. http://software.intel.com/en-us/articles/intel-energy-checker-sdk/

Q5 - Why use the Intel® Energy Checker SDK? What is this all about?

A5 - The software industry lacks clear metrics that correlate business productivity achieved and the amount of energy consumed to achieve it. The software ecosystem lacks the ability to measure energy consumed by software, and the Intel® Energy Checker SDK (Intel® EC SDK) has been created to provide a solution for these issues. With the Intel® EC SDK, ISVs can instrument their applications to export measures of "useful work" performed and correlate that useful work with energy consumed by the system. Through an iterative process, developers can write energy-aware software. At a facility or service level, individual measures of useful work can be aggregated to drive automated policies that dynamically balance service level agreements with energy consumption.

Q6 - How much does the Intel® Energy Checker SDK cost?

A6 - Nothing. The Intel® Energy Checker SDK is available as a no-cost, royalty-free licensed download from Intel at http://software.intel.com/en-us/articles/intel-energy-checker-sdk/

Q7 - Is this SDK only for servers?

A7 - No. Although it was initially developed for a use in the data center and telecom environments, the SDK can be used on client or mobile platforms.

Q8 - What are some example use cases for the Intel® Energy Checker SDK?

A8 - There are many possible use cases for the Intel® Energy Checker SDK, but three of the most common usages are described below.

1. Lab Usage
   Engineers can evaluate the most efficient configuration of hardware and software components by comparing the useful work performed vs. the energy consumed. Engineers can also evaluate different algorithms or work flows to see which algorithm provides the highest efficiency.
2. Production-level Data Center Long-term Trending
   There is an engineering maxim that *you can't manage what you can't measure*, and this is true for managers of data centers and other IT facilities. Actual application usage may deviate from usage patterns predicted in the lab or at initial system deployment. The lightweight counter instrumentation available using the Intel® EC SDK enables data center managers to track software activity in deployed environments, identify trends, and justify new purchases.
3. Real-time Dynamic Management
   Taking the production-level trending concept one step further, the Intel® EC SDK can be used to provide real-time management of applications. For example, if a service is provided on a cluster of servers and aggregated PL data from multiple servers indicates that there is little utilization of the cluster at the current time, management software could decide to quiesce extra servers to save power. Alternatively, the management software could choose to reduce the frequency of processors in those servers.

More information on these use cases can be found in the white paper at http://software.intel.com/en-us/articles/intel-energy-checker-sdk/.

Q9 - Who should be interested in the Intel® Energy Checker SDK?

A9 - Software developers, IT and data center managers, and line of business managers should all be interested in the Intel® EC SDK.

Q10 - Why should software developers be interested in the Intel® Energy Checker SDK?

A10 - The value of more efficient software is often overlooked by managers overly focused on hardware speeds and feeds. The Intel® Energy Checker SDK provides software developers a simple API to expose metrics of "useful work" done by an application through easy software instrumentation. The amount of useful work done by a payroll application is different from the amount of useful work performed by a video serving application, a database application, or a mail server application. All too often, activity is measured by how *busy* a server is while running an application rather than by *how much* work that application completes. The Intel® EC SDK provides a way for the software developer to determine what measures of "useful work" are important *for that application* and expose those metrics through a simple API. Software developers can improve the energy efficiency of their code by correlating their application performance with the energy consumption of a system using the energy monitoring tools provided with the SDK.

Software developers can also readily expose key conditions to watch, such as the number of outstanding requests in a queue. These can provide critical clues to application performance and potential bottlenecks that administrators should watch over time.

The Intel® EC SDK provides basic tools to support logging and manual analysis of useful work, but there is also an opportunity for management application developers to aggregate measures of useful work taken at the application level, middleware level, and OS/hardware level; based on this information, the management application can direct automated policies such as demand-driven virtual machine migration.

Q11 - Why should IT and data center managers be interested in the Intel® Energy Checker SDK?

A11 - IT and data center managers have traditionally been concerned with how to provide the most cost-effective hardware and software configurations for their data centers. They have often relied on standard benchmarks as a rough analog to the performance they would expect to see with the various applications they deploy because many applications lack easily accessed metrics indicating application performance and key bottleneck indicators. As applications instrument both their measures of "useful work" and their application watch conditions, data center managers can better ascertain what hardware configurations will work best for a given configuration and what software configurations/combinations will give the best performance. More importantly, the Intel® EC SDK is light enough to be used in a production environment so that those initial conclusions can be re-examined with actual workloads and adjustments can be made over time if necessary.

As energy costs have become a more critical factor in Total Cost of Ownership (TCO), the energy monitoring tools in the Intel® EC SDK provide ways for IT and data center managers to better track the energy consumption of specific servers (or racks/rows of servers); this can be used to bill back specific departments for given services or, coupled with information tracking the "useful work" of applications, can help support/defend new CapEx and OpEx budget requests.

Q12 - Why should Lines of Business Managers be interested in the Intel® Energy Checker SDK?

A12 - Lines of Business Managers (Telcos, Hosting Service providers, and Cloud operators) are responsible for defining the ROI (Return on Investment) for the services they deploy. The "useful work" metrics implemented by the software developers provide these managers with the raw data to show how and when an application is being used.

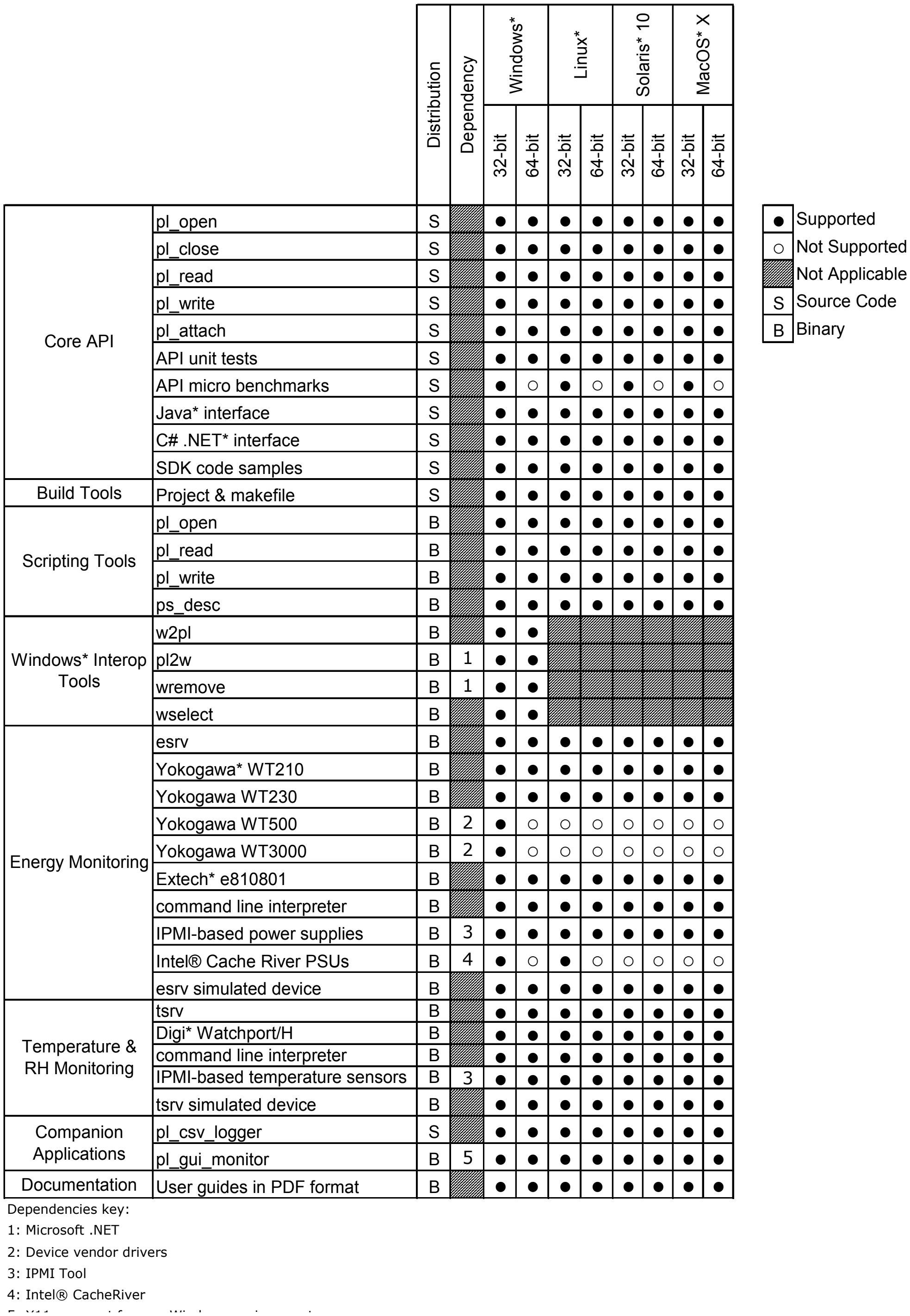
In addition, there is considerable pressure on industry today to "go green". By correlating the amount of useful work performed by applications vs. the power consumption of those systems and the related infrastructure, managers can track efficiency gains. By basing technology selections on performance and efficiency, line of business managers will incent suppliers to produce and deploy more efficient hardware and software. This improves both the company's *perception* of being green and the company's *actual performance* in becoming greener, while delivering the added benefit of reducing power bills over the long term.

Q13 - Are there standard measures of "useful work"?

A13 - There is no universal quantum of useful work, because application needs vary widely. As similar applications instrument their code with the Intel® Energy Checker SDK, we anticipate the industry will develop some standard measures of useful work. For example, telecom applications handling SMS (Short Message Service) text messages would likely have counters for the number of text messages sent and received. The industry will collectively determine what measures of useful work make sense for specific applications; this may evolve similarly to how the networking industry defined common MIBs (Management Information Bases) for network equipment using SNMP (Simple Network Management Protocol).

Q14 - What operating systems are supported? What is included in the Intel® Energy Checker SDK?

A14 - Operating systems supported include Windows*, Linux*, Solaris* 10, and MacOS X*. The Intel® EC SDK includes three manuals (SDK User Guide, SDK Companion Applications User Guide, and SDK Device Kit User Guide), source code for the core API, and numerous utilities. Some utilities are in binary form only, while others are provided as source code, as shown in the figure below:



		Distribution	Dependency	Windows*		Linux*		Solaris* 10		MacOS* X	
				32-bit	64-bit	32-bit	64-bit	32-bit	64-bit	32-bit	64-bit
Core API	pl_open	S	▨	●	●	●	●	●	●	●	●
	pl_close	S	▨	●	●	●	●	●	●	●	●
	pl_read	S	▨	●	●	●	●	●	●	●	●
	pl_write	S	▨	●	●	●	●	●	●	●	●
	pl_attach	S	▨	●	●	●	●	●	●	●	●
	API unit tests	S	▨	●	●	●	●	●	●	●	●
	API micro benchmarks	S	▨	●	○	●	○	●	○	●	○
	Java* interface	S	▨	●	●	●	●	●	●	●	●
	C# .NET* interface	S	▨	●	●	●	●	●	●	●	●
	SDK code samples	S	▨	●	●	●	●	●	●	●	●
Build Tools	Project & makefile	S	▨	●	●	●	●	●	●	●	●
Scripting Tools	pl_open	B	▨	●	●	●	●	●	●	●	●
	pl_read	B	▨	●	●	●	●	●	●	●	●
	pl_write	B	▨	●	●	●	●	●	●	●	●
	ps_desc	B	▨	●	●	●	●	●	●	●	●
Windows* Interop Tools	w2pl	B	▨	●	●	▨	▨	▨	▨	▨	▨
	pl2w	B	1	●	●	▨	▨	▨	▨	▨	▨
	wremove	B	1	●	●	▨	▨	▨	▨	▨	▨
	wselect	B	▨	●	●	▨	▨	▨	▨	▨	▨
Energy Monitoring	esrv	B	▨	●	●	●	●	●	●	●	●
	Yokogawa* WT210	B	▨	●	●	●	●	●	●	●	●
	Yokogawa WT230	B	▨	●	●	●	●	●	●	●	●
	Yokogawa WT500	B	2	●	○	○	○	○	○	○	○
	Yokogawa WT3000	B	2	●	○	○	○	○	○	○	○
	Extech* e810801	B	▨	●	●	●	●	●	●	●	●
	command line interpreter	B	▨	●	●	●	●	●	●	●	●
	IPMI-based power supplies	B	3	●	●	●	●	●	●	●	●
	Intel® Cache River PSUs	B	4	●	○	●	○	○	○	○	○
	esrv simulated device	B	▨	●	●	●	●	●	●	●	●
Temperature & RH Monitoring	tsrv	B	▨	●	●	●	●	●	●	●	●
	Digi* Watchport/H	B	▨	●	●	●	●	●	●	●	●
	command line interpreter	B	▨	●	●	●	●	●	●	●	●
	IPMI-based temperature sensors	B	3	●	●	●	●	●	●	●	●
	tsrv simulated device	B	▨	●	●	●	●	●	●	●	●
Companion Applications	pl_csv_logger	S	▨	●	●	●	●	●	●	●	●
	pl_gui_monitor	B	5	●	●	●	●	●	●	●	●
Documentation	User guides in PDF format	B	▨	●	●	●	●	●	●	●	●

●	Supported
○	Not Supported
▨	Not Applicable
S	Source Code
B	Binary

Dependencies key:
1: Microsoft .NET
2: Device vendor drivers
3: IPMI Tool
4: Intel® CacheRiver
5: X11 support for non-Windows environment

Q15 - Can I review the API source code?

A15 - Yes, the core API and several SDK samples are shipped as source code in the Intel® Energy Checker SDK. This not only allows you to review the source code, but it permits you to integrate the API code as-is into your source tree if you wish. The other reason for this choice is to allow you to build the API according to your application’s needs (built-in, shared library, static library, 32-bit executable, 64-bit executable, etc.).

Q16 - Is the code open source?

A16 - No, the Intel® Energy Checker SDK is not distributed under an open source license. However, the Intel® Energy checker SDK is free of charge and its core API source code is provided so it can be reviewed. Please read the license agreement for details; it can be accessed here (http://software.intel.com/en-us/articles/what-if-pre-release-license-agreement/).

Q17 - Can I modify the API source code?

A17 - It depends. If it is for an in-house or experimental use, then you can modify the source code; if you find a useful modification that you think would be useful to others, please let us know through the What If Software Forum. However, you cannot use a modified version of the API in an application that you distribute outside your organization or that you plan to deploy in production. We impose this restriction so counter interoperability is maintained. The value of Intel® Energy Checker instrumentation grows as more software and more levels of the system software stack implement this API, but consistent APIs are required for end-users to capitalize on this value. Consistent API usage also enables third-party management tools and middleware to build upon the counters exported by different applications.

Q18 - How can I be sure that the API works if I make modifications for test use?

A18 - The Intel® Energy Checker SDK is shipped with a unit test for this purpose. Please read §3.12 of the Intel® Energy Checker User Guide for more details.

Q19 - What is covered in the Intel® Energy Checker SDK User Guide?

A19 - The Intel® Energy Checker SDK User Guide provides a general overview of the SDK, describes the core API, describes how to install the SDK and build code with the SDK, describes the scripting tools and interoperability tools for the Windows registry, provides brief information on the ESRV and TSRV utilities, reviews the sample utilities provided with the SDK, and concludes with information showing how to access the SDK counter information across a network.

Q20 - What is covered in the Intel® Energy Checker SDK Device Kit User Guide?

A20 - The Intel® Energy Checker SDK Companion Applications User Guide describes two key applications provided with the Intel® Energy Checker SDK and how they can be extended to support additional devices:

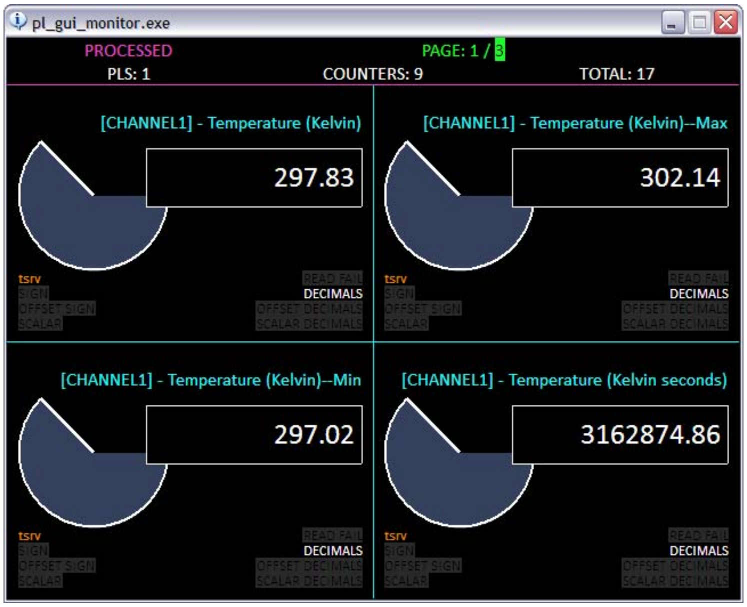
1. The ESRV energy monitoring tool measures energy consumption and power usage through the use of instrumented servers or external power meters.
2. The TSRV temperature monitoring tool measures temperature and relative humidity via supported sensors. Energy efficiency results with similar systems can vary depending on critical environmental parameters, and the TSRV application helps baseline conditions for comparative purposes and long-term trending analysis.

ESRV and TSRV have similar structures and command lines to make it easier for developers and end-users to integrate energy and temperature sensing into their applications, thereby facilitating the development of energy-aware software. Both of these utilities allow developers to extend the list of supported devices through the use of device libraries.

Q21 - What is covered in the Intel® Energy Checker SDK Companion Applications User Guide?

A21 - The Intel® Energy Checker SDK Companion Applications User Guide describes two companion applications provided with the Intel® Energy Checker SDK:

1. PL GUI Monitor provides a graphical tool which allows users to monitor one or more sets of counters simultaneously in real time. The PL GUI Monitor can be used for building IT dashboards, studying applications' performance, monitoring systems' behaviors, or debugging applications. One example screenshot is shown below:



pl_gui_monitor.exe

PROCESSED PAGE: 1 / 3

PLS: 1 COUNTERS: 9 TOTAL: 17

[CHANNEL1] - Temperature (Kelvin) 297.83

[CHANNEL1] - Temperature (Kelvin)--Max 302.14

[CHANNEL1] - Temperature (Kelvin)--Min 297.02

[CHANNEL1] - Temperature (Kelvin seconds) 3162874.86

2. PL CSV Logger is a console application to log one or more sets of counters into a Comma Separated Value (CSV) text file. PL CSV Logger was created as both a tool and as a reference application (source code is provided for PL CSV Logger).

Q22 - Why are the main API files called productivity_link.h and productivity_link.c?

A22 - This reflects the earlier name for the Intel® Energy Checker SDK. As noted below, a PL contains a collection of productivity counters. These counters can be linked to energy consumption to determine overall efficiency.

Q23 - What is a counter? What is a PL?

A23 - A counter is a stored value used to record application activity. For example, a mail application may define counters for the number of e-mails sent, number of e-mails received, number of bytes sent, and the number of bytes received, among other counters. These counters are stored as unsigned 64-bit values and usually represent activity that increases over time (similar to an odometer in a car). Counters are always stored as unsigned 64-bit integers to keep the API lightweight and extremely efficient.

A PL (formerly called a productivity link) is a collection of one or more counters. Most applications will only have a single PL but may have many counters in that PL; some applications may also have multiple PLs. An application can either re-use an existing PL each time it is invoked, or it can create a new PL each time it is started.

Q24 - Why are counters unsigned values? Can I represent negative numbers with a counter?

A24 - Unsigned values provide the best portability between different operating systems and processor architectures. Negative numbers are sometimes treated differently internally on dissimilar systems, but unsigned values are readily transportable between different systems.

Negative values can be represented with the *.sign* suffix counter.

Q25 - What are suffix counters?

A25 - Suffix counters are supplemental counters that help other software interpret the meaning behind a given (base) counter. Each of these suffix counters is handled like a regular counter and not interpreted by the Intel® Energy Checker API, but applications like ESRV, TSRV, and PL GUI Monitor follow a convention to make use of counters named with the following suffixes appended to the base counter's name:

- *.sign* is non-zero if the base counter is negative
- *.decimals* indicates the number of decimal places to apply (3 = divide by 10^3)
- *.scalar* indicates a constant value by which to multiply the base counter
- *.offset* indicates a constant value to add to the base counter

These and other suffix counters are described in §2.3.4 of the Intel® EC User Guide.

Q26 - How many counters can an application define? How frequently can they be updated?

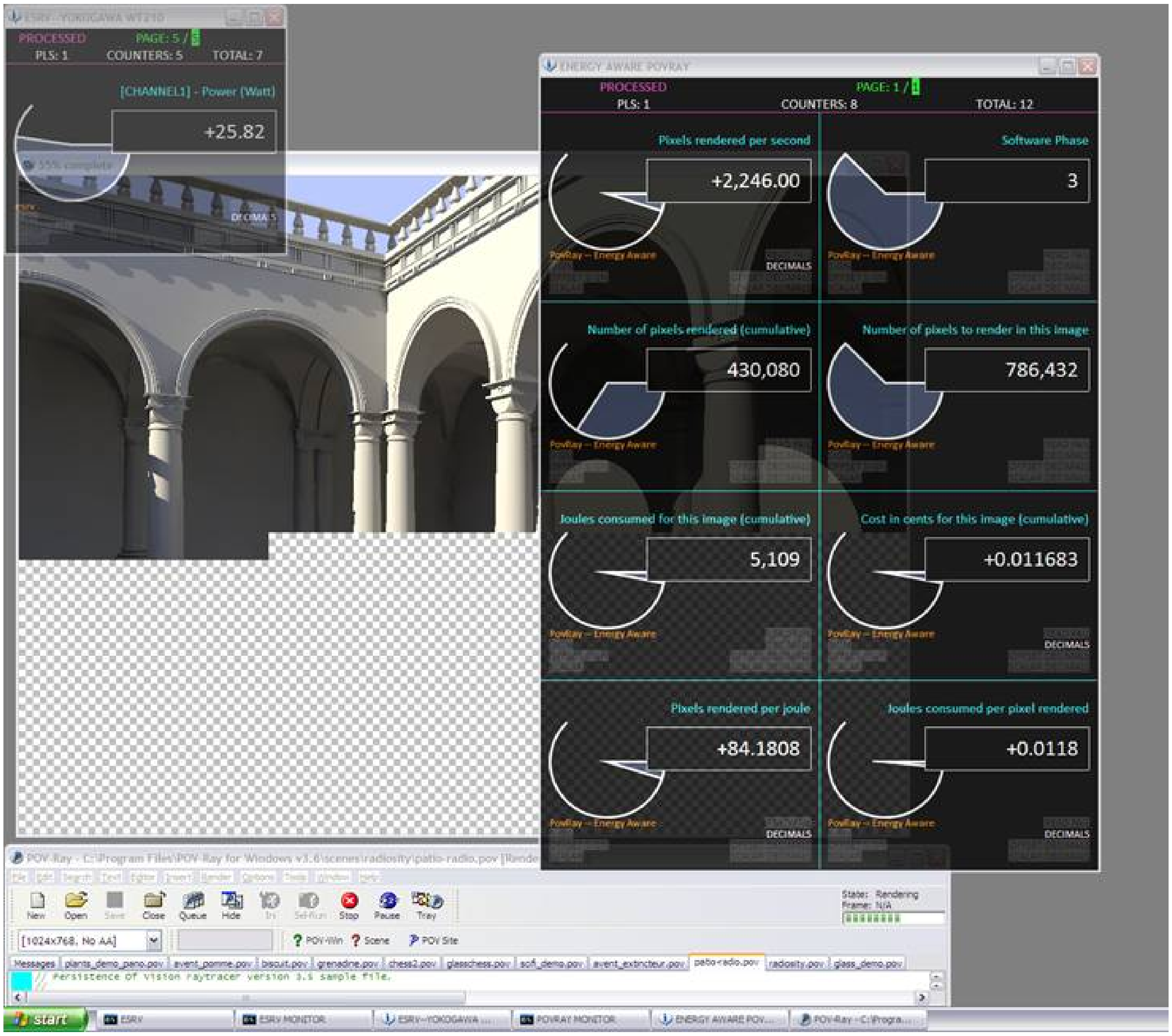
A26 - Most applications will define less than 50 counters that they export, but some applications could have over 1000 counters, all at the discretion of the application developer. Some counters may be updated only once per day, while other counters may be updated once per second. Developers are encouraged to write each counter no more than one time per second and for each application to write no more than 200 counter updates per second, but this is at the discretion of the developer.

Q27 - **How intrusive is the instrumentation?**

A27 - The performance impact is negligible. Even a relatively slow dual-core system with one processor doing 1000 counter updates per second should see less than 0.5% performance impact. Faster processor cores and larger numbers of cores/processors will reduce this even further. To keep the impact low, we recommend updating these counters at a maximum rate of one update per second. The SDK is shipped with a micro-benchmark that allows you to evaluate the impact of Intel® EC API pl_read and pl_write calls on your system; please read §3.13 of the Intel® Energy Checker SDK User Guide for details. Note also that the API can be compiled with the __PL_BYPASS__ symbol defined to gracefully fail all calls to the API; testing code with and without this flag enabled can help a developer gauge the instrumentation impact on their application's performance.

Q28 - **Is there a tool to monitor counters or PLs in the SDK?**

A28 - Yes, the PL GUI Monitor companion tool is the tool you are looking for. Please read the Intel® Energy Checker SDK Companion Application User Guide for details. The figure below demonstrates the use of PL GUI Monitor to display the ESRV counters (top left window) and an energy-aware private version of the POV-Ray* graphics rendering software.


ESRV--YOKOGAWA WT210

PROCESSED PAGE: 5 / 5

PLS: 1 COUNTERS: 5 TOTAL: 7

[CHANNEL1] - Power (Watt): +25.82

ENERGY AWARE POVRAY

PROCESSED PAGE: 1 / 1

PLS: 1 COUNTERS: 8 TOTAL: 12

Pixels rendered per second: +2,246.00

Software Phase: 3

Number of pixels rendered (cumulative): 430,080

Number of pixels to render in this image: 786,432

Joules consumed for this image (cumulative): 5,109

Cost in cents for this image (cumulative): +0.011683

Pixels rendered per joule: +84.1808

Joules consumed per pixel rendered: +0.0118

POV-Ray - C:\Program Files\POV-Ray for Windows v3.6\scenes\radiosity\patio-radio.pov [Rende

State: Rendering

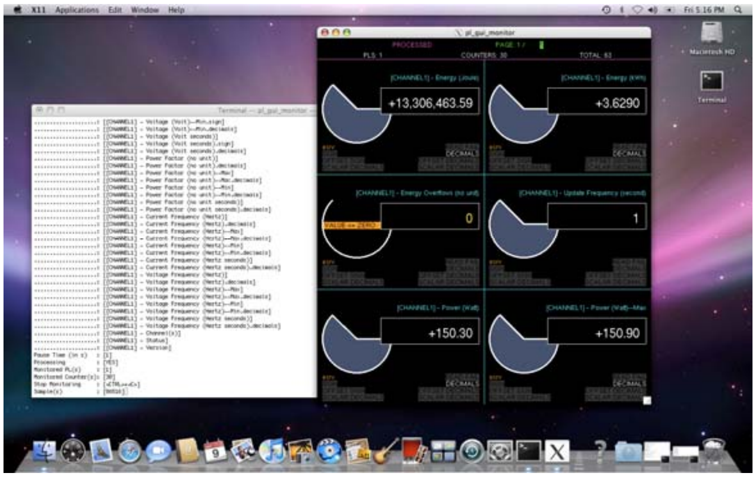
Frame: N/A

[1024x768, No AA]

// Persistence of Vision Raytracer version 3.5 sample file.

Q29 - Does PL GUI Monitor work on UNIX*-like systems?

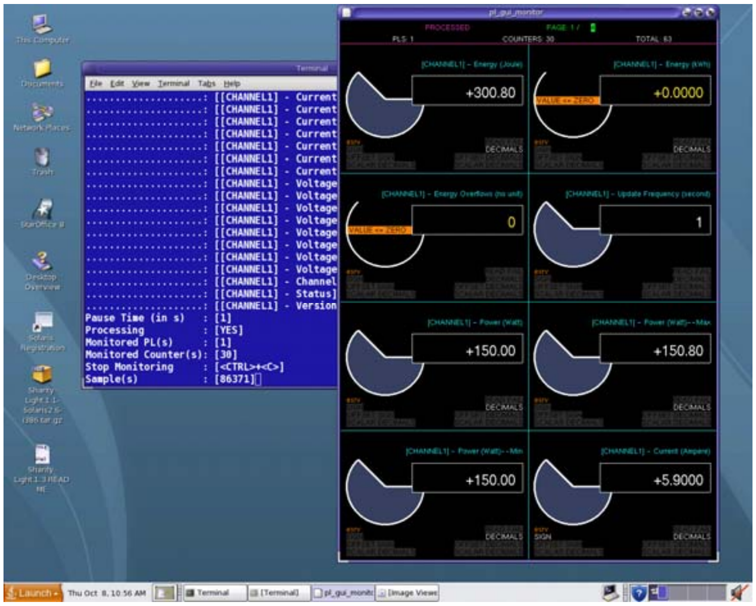
A29 - Yes, the PL GUI Monitor works with X11*. There are slight cosmetic differences between the Windows* and the UNIX* versions, but they are functionally identical. The figures below show the PL GUI Monitor running on MacOS* X, Solaris* 10 and Linux*. Please note also that you can remotely monitor PL counters.



X11 Applications Edit Window Help
Fri 5:16 PM
pl_gui_monitor
PROCESSED
PAGE 1 /
PLS: 1
COUNTERS: 30
TOTAL: 63
[CHANNEL1] - Energy (Joule)
+13,306,463.59
[CHANNEL1] - Energy (kWh)
+3.6290
[CHANNEL1] - Energy Overflows (no unit)
0
VALUE <= ZERO
[CHANNEL1] - Update Frequency (second)
1
[CHANNEL1] - Power (Watt)
+150.30
[CHANNEL1] - Power (Watt)--Max
+150.90
DECIMALS
Macintosh HD
Terminal
Terminal — pl_gui_monitor

```
....................: [[CHANNEL1] - Voltage (Volt)--Min.sign]
....................: [[CHANNEL1] - Voltage (Volt)--Min.decimals]
....................: [[CHANNEL1] - Voltage (Volt seconds)]
....................: [[CHANNEL1] - Voltage (Volt seconds).sign]
....................: [[CHANNEL1] - Voltage (Volt seconds).decimals]
....................: [[CHANNEL1] - Power Factor (no unit)]
....................: [[CHANNEL1] - Power Factor (no unit).decimals]
....................: [[CHANNEL1] - Power Factor (no unit)--Max]
....................: [[CHANNEL1] - Power Factor (no unit)--Max.decimals]
....................: [[CHANNEL1] - Power Factor (no unit)--Min]
....................: [[CHANNEL1] - Power Factor (no unit)--Min.decimals]
....................: [[CHANNEL1] - Power Factor (no unit seconds)]
....................: [[CHANNEL1] - Power Factor (no unit seconds).decimals]
....................: [[CHANNEL1] - Current Frequency (Hertz)]
....................: [[CHANNEL1] - Current Frequency (Hertz).decimals]
....................: [[CHANNEL1] - Current Frequency (Hertz)--Max]
....................: [[CHANNEL1] - Current Frequency (Hertz)--Max.decimals]
....................: [[CHANNEL1] - Current Frequency (Hertz)--Min]
....................: [[CHANNEL1] - Current Frequency (Hertz)--Min.decimals]
....................: [[CHANNEL1] - Current Frequency (Hertz seconds)]
....................: [[CHANNEL1] - Current Frequency (Hertz seconds).decimals]
....................: [[CHANNEL1] - Voltage Frequency (Hertz)]
....................: [[CHANNEL1] - Voltage Frequency (Hertz).decimals]
....................: [[CHANNEL1] - Voltage Frequency (Hertz)--Max]
....................: [[CHANNEL1] - Voltage Frequency (Hertz)--Max.decimals]
....................: [[CHANNEL1] - Voltage Frequency (Hertz)--Min]
....................: [[CHANNEL1] - Voltage Frequency (Hertz)--Min.decimals]
....................: [[CHANNEL1] - Voltage Frequency (Hertz seconds)]
....................: [[CHANNEL1] - Voltage Frequency (Hertz seconds).decimals]
....................: [[CHANNEL1] - Channel(s)]
....................: [[CHANNEL1] - Status]
....................: [[CHANNEL1] - Version]
Pause Time (in s)   : [1]
Processing          : [YES]
Monitored PL(s)     : [1]
Monitored Counter(s): [30]
Stop Monitoring     : [<CTRL>+<C>]
Sample(s)           : [[illegible]]
```

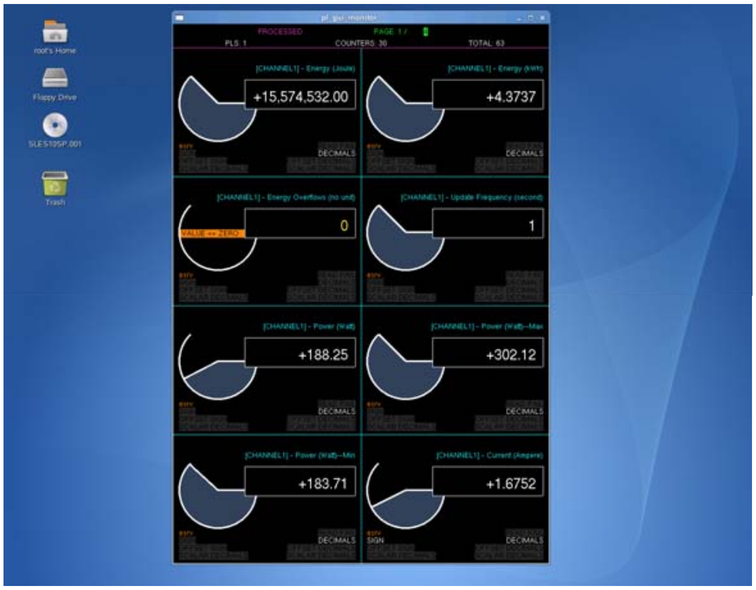
PL GUI Monitor running on MacOS* X



Terminal
File Edit View Terminal Tabs Help

```
....................: [[CHANNEL1] - Current
....................: [[CHANNEL1] - Current
....................: [[CHANNEL1] - Current
....................: [[CHANNEL1] - Current
....................: [[CHANNEL1] - Current
....................: [[CHANNEL1] - Current
....................: [[CHANNEL1] - Current
....................: [[CHANNEL1] - Voltage
....................: [[CHANNEL1] - Voltage
....................: [[CHANNEL1] - Voltage
....................: [[CHANNEL1] - Voltage
....................: [[CHANNEL1] - Voltage
....................: [[CHANNEL1] - Voltage
....................: [[CHANNEL1] - Voltage
....................: [[CHANNEL1] - Voltage
....................: [[CHANNEL1] - Channel
....................: [[CHANNEL1] - Status]
....................: [[CHANNEL1] - Version
Pause Time (in s)   : [1]
Processing          : [YES]
Monitored PL(s)     : [1]
Monitored Counter(s): [30]
Stop Monitoring     : [<CTRL>+<C>]
Sample(s)           : [86371]
```

PL GUI Monitor running on Solaris* 10



PLS: 1 COUNTERS: 30 TOTAL: 63

[CHANNEL1] - Energy (Joule) +15,574,532.00

[CHANNEL1] - Energy (kWh) +4.3737

[CHANNEL1] - Energy Overflows (no unit) 0

[CHANNEL1] - Update Frequency (second) 1

[CHANNEL1] - Power (Watt) +188.25

[CHANNEL1] - Power (Watt)--Max +302.12

[CHANNEL1] - Power (Watt)--Min +183.71

[CHANNEL1] - Current (Ampere) +1.6752

PL GUI Monitor running on Linux*

Q30 - What benefits can I expect from using Intel® Energy Checker SDK?

A30 - By using the Intel® Energy Checker SDK, you will be able to define and dynamically measure your application's energy efficiency while it is running on your hardware and is processing your data set(s). With minimal extra effort, you could also turn your software into an energy-aware application able to expose its energy efficiency metrics and to take coordinated or autonomous actions to adapt its behavior according to some energy heuristics. Armed with this knowledge you could optimize your application's energy efficiency by using a standard iterative methodology described in the Intel® EC SDK User Guide. More interestingly, your customers could use this data to assess *in situ* the efficiency of their IT equipment and better plan/manage it. Last but not least, by exposing energy efficiency metrics using the Intel® EC API, management middle-ware or intelligent hardware could use them to implement software energy efficiency rules. Your imagination is the limit!

Q31 - How do you define software energy efficiency?

A31 - Software energy efficiency is loosely defined as the ratio of the amount of useful work performed by the software and the energy consumed by the hardware to produce this work. For example, if the useful work performed by an application is the number of client requests processed, then a possible energy efficiency metric could be the number of client requests processed per joule consumed – or vice-versa. Because the amount of useful work is application specific, developers have the total freedom and expertise to define it. The Intel® EC SDK is shipped with a tool named ESRV which uses the Intel® EC API to expose the amount of energy consumed by monitored system(s). Therefore, any application using the API can correlate the system's useful work with the system's energy consumption. Please read chapter eight in the Intel® Energy Checker SDK User Guide for more details.

Q32 - Why energy? Why not performance per Watt?

A32 - Strictly speaking, electronics consume energy, not power. If you know precisely the time elapsed during an analysis, then you can convert between energy and average power.

Q33 - What is the difference between power and energy?

A33 - Power is a rate of energy consumption, while energy is a measure of power over time. Electricity is billed in kiloWatt-hours (kW-h) based on energy consumed. Average power can be determined by dividing energy consumption by the elapsed time.

The ESRV utility provided with the Intel® EC SDK provides both instant power (in Watts and Joules) and elapsed energy consumption (in kW-h). Please read §8.2 of the Intel® Energy Checker SDK User Guide for details.

Q34 - What is a Joule?

A34 - A Joule is a measure of energy equal to one Watt-second. Since there are 3600 seconds in an hour and 1000 Watts in a kiloWatt, there are 3.6 million Joules in a kiloWatt-hour.

Q35 - Is energy-aware software very different from a simply work-instrumented one?

A35 - Not at all. The sole difference is that the energy-aware application imports the energy counters of an ESRV instance monitoring the system(s) on which it runs. The figure below shows the pseudo-code difference between work-instrumented and energy-aware applications.

Work-instrumented Application

```
Initialize {

      pl_open for App PL
}
For each work unit {



      pl_write to App PL
            work done

}
Clean up {
      pl_close App PL

}
```

Energy-aware Application

```
Initialize {
      pl_attach to ESRV PL
      pl_open for App PL
}
For each work unit {
      pl_read from ESRV PL
            Energy
      compute efficiency
      pl_write to App PL
            work done
            efficiency data
}
Clean up {
      pl_close App PL
      pl_close ESRV PL
}
```

Q36 - Do I really have to measure energy in my application?

A36 - No, simply by exposing the amount of useful work in your application, you enable third-party tools to correlate the work and energy data to derive energy efficiency metrics for your software. For example, the PL CSV Logger companion/sample application of the Intel® EC SDK could be used to capture counters from your application and ESRV in a CSV file. A spreadsheet application can be used later to compute energy efficiency metrics based on the logged data.

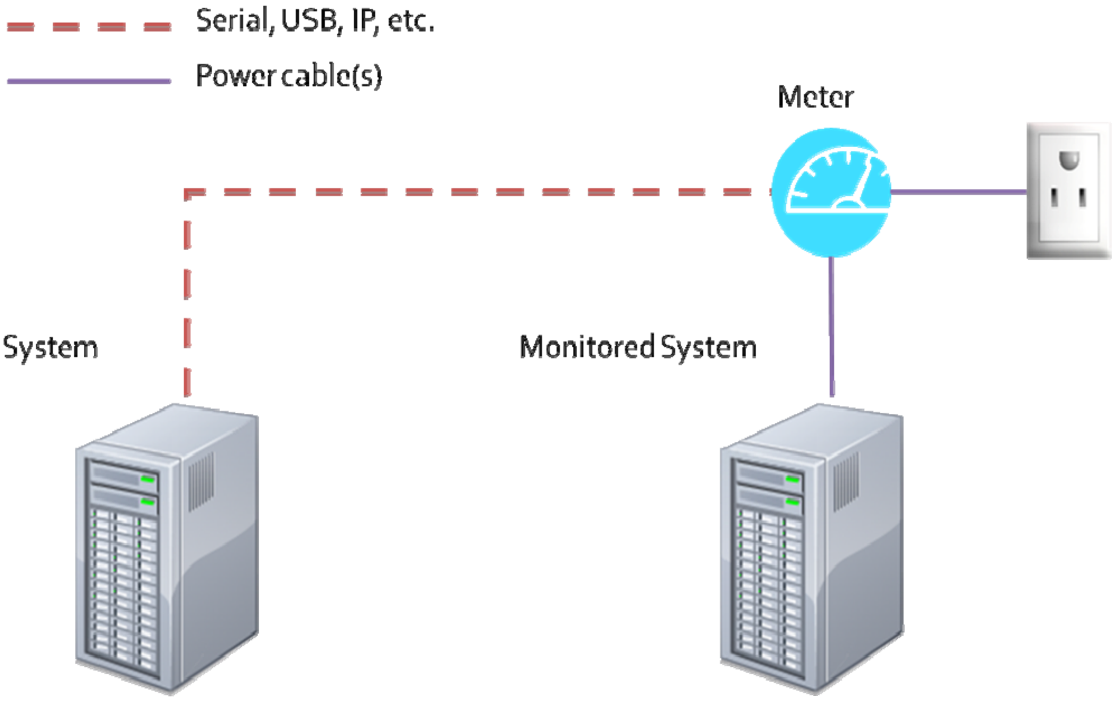
In addition to accounting for the work done, we strongly recommend exporting a counter that reports the software phase of the application. For example, an application may have distinct phases for startup, standby, active, maintenance, and shutdown modes. Some efficiency metrics may apply only during the active mode, so a software phase counter that provides an ordinal value corresponding to the current software phase could help analysis software better characterize the efficiency of a given system.

Q37 - Do I need any extra equipment to measure the energy consumed?

A37 - This depends on the server (or servers) you are monitoring. Some servers have internal instrumentation to measure the server's power draw, but many servers do not have this feature; if the equipment does not contain input power sensors, you will need a power analyzer. A power analyzer/reader measures equipment power draw.

The Intel® Energy Checker SDK is shipped with a tool named ESRV; ESRV is capable of driving several power readers available on the market. ESRV can also use the open-source IPMITool* utility to read power information from servers instrumented with IPMI. For additional device support, ESRV can interpret data from many command-line utilities; alternatively, power analyzer vendors can provide support libraries to interface their meters directly to ESRV.

With an external power analyzer, the ESRV utility can run on the same system as the system being monitored, or it can run on a separate system, as shown in the figure below.



Keep in mind that while a power analyzer will generally measure power at the wall (AC input power); some power supply sensors may measure power behind the power supply (DC output power). Measuring just the DC output ignores losses in the power supply and could be off by as much as 30%. By decoupling the energy measurement from the workload measurement, the end-user is free to select internal or external power metering with ESRV without affecting the application code instrumented with the Intel® EC SDK.

Q38 - What external power analyzers are supported?

A38 - ESRV is shipped with support for the Extech* 380801 and the Yokogawa* WT210, WT500, WT230, and WT3000 external digital power readers.

Image credits: Images courtesy of Extech and Yokogawa.

Q39 - How can other devices be supported?

A39 - ESRV defines a standard interface to allow power reader vendors and system integrators to offer support for their devices. This interface is detailed in the Intel® Energy Checker Device Kit User Guide. Vendors wishing to add support for their power readers must provide a dynamic link library for Windows and shared object files for Linux, Solaris, and MacOS X. Vendors are encouraged to provide support for the entire range of OS types, as well as both 32- and 64-bit versions of ESRV.

The library must implement the six functions listed below:

```
//-----------------------------------------------------------------------------
// functions prototype
//-----------------------------------------------------------------------------
ESRV_API int init_device_extra_data(PESRV);
ESRV_API int delete_device_extra_data(PESRV);
ESRV_API int open_device(PESRV, void *);
ESRV_API int close_device(PESRV, void *);
ESRV_API int parse_device_option_string(PESRV, void *);
ESRV_API int read_device_power(PESRV, void *, int);
```

If the device supports hardware integration of power to compute energy, then the following function should be implemented in the library:

```
ESRV_API int read_device_energy(PESRV, void *, int, int);
```

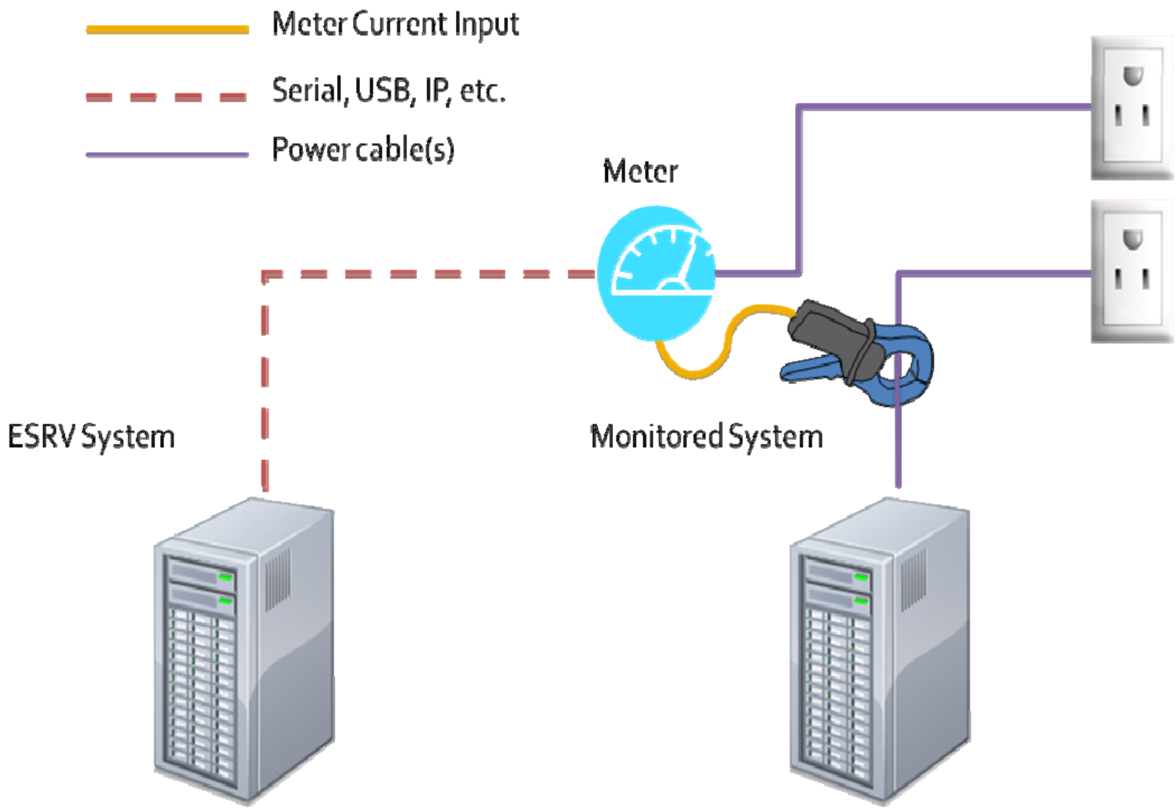
The SDK is shipped with template device code that can be used to build the device library.

Q40 - How can I add my company's power (or temperature) reader to the SDK supported devices list?

A40 - We defined a simple interface to the ESRV and TSRV tools in the SDK to measure energy consumption and the ambient temperature. (See the answer to Q39 - this lists the library functions required.) The SDK also provides commented sample libraries that you can use as a starting point. Note that we encourage you to provide a support library for each supported operating system (Windows*, Linux*, Solaris 10* and MacOS X*), in 32- and 64-bit versions. Once your libraries are done, please contact us via the What If Software Forum to suggest adding a reference to your device support libraries.

Q41 - **Do I have to unplug my production system(s) to measure energy?**

A41 - No, you can use a clamp-on probe as described in chapter two of the Intel® Energy Checker Device Kit User Guide. Refer to the figure in A37 - this demonstrates a setup where the monitored system is electrically connected to the power meter; this requires stopping and unplugging the system. The figure below demonstrates the use of a clamp-on probe in a setup where the monitored system is not connected electrically to the power meter, and therefore doesn't require unplugging the system.



Q42 - **Do I need to install any driver?**

A42 - No.

Q43 - **Can I break down the energy consumed at the component level (CPU, memory, etc.)?**

A43 - Not in this version of the SDK. Future versions of ESRV may allow you to do finer-grained energy analysis, such as measuring DIMM energy, processor energy, etc. This capability will likely require the use of Data-Acquisition (DAQ) equipment and an instrumented board. Please check the WhatIf.intel.com for future updates of the Intel® Energy Checker SDK.

Q44 - **Is it a problem if multiple applications are running on the same system?**

A44 - No, the Intel® Energy Checker API ensures that no data collisions happen when multiple applications are running on the same system [unless the software explicitly re-uses the same PL each time it loads]. This extends to multiple instances of the same application.

Q45 - Is it a problem my application is running on multiple systems?

A45 - No, you should measure all the hardware used to generate the amount of useful work. For example, if your application is distributed over five servers, uses a NAS and has a dedicated switch, then the energy consumed by all this equipment should be correlated to the amount of useful work done.

Energy consumption of multiple pieces of equipment can be measured using several methods. For example, the power measurement can be aggregated at the power meter level by using special power strips or using multiple channels in a multi-channel power analyzer. Energy can also be measured by aggregating data from different instances of ESRV (running on a single system or running on multiple servers). As always, some deployments may require a special setup.

Q46 - Can I use the SDK in a virtualized environment?

A46 - Yes, there are no major differences between a virtualized system with multiple VMs (Virtual Machines) and multiple servers running on physically separate boxes--other than the fact that these VMs may all run off the same power supply. Correlating energy consumption with useful work when multiple VMs run on the same system is similar to correlating energy measured from a power strip to useful work from multiple physical servers.

In a virtualized environment, each virtual machine looks like a different system. If remote data collection is desired, proper network or system configuration is required for to aggregate data from each of the VMs. Please read the Annex of the Intel® EC SDK User Guide for an example of network setup.

Q47 - If multiple applications are running on my system, what does it mean from an energy efficiency point of view?

A47 - To simplify management, many administrators run homogeneous workloads on a given server rather than running dissimilar services on the same server. If this approach is taken, then it is possible – and meaningful – to build a composite work metric such as *client requests served and MB of data extracted from a database*. In other cases, more complex metrics may need to be developed in order to apportion energy consumption to a given application or workload.

Q48 - How do I get started?

A48 - Download and install the SDK on your development system (http://software.intel.com/en-us/articles/intel-energy-checker-sdk/). Please read chapter four of the Intel® Energy Checker SDK User Guide for details. Also read chapters one and two to get an overview of what this SDK is and acquire basic SDK lingo. It is then time to instrument your application. As a first step, you should spend some time to define the useful work performed by your program. This can be anything that is meaningful to you or your customers (frames encoded, pixels rendered, KB of mail sent, client requests served, etc.). This is likely the most important step in the instrumentation process. Then use the esrv_client sample code (also presented as pseudo C code in chapter eight of the Intel® Energy Checker SDK User Guide) as a template to instrument your application.

Q49 - Is there a *Hello World* sample in the SDK?

A49 - No, but the C code below is an example of a very simple application using the Intel® EC API:

```
#include ”productivity_link.h”
void main(void) {
        uuid_t uuid;
        unsigned long long the_answer = 42;
        int pld = PL_INVALID_DESCRIPTOR;
        char *counter[] = { ”World” };
        if((pld = pl_open(”Hello”, 1, counter, &uuid)) != PL_INVALID_DESCRIPTOR) {
                pl_write(pld, &the_answer, 0);
                pl_close(pld);
        }
}
```

This writes a value of 42 to the counter named ”World” in a new PL for the application named ”Hello”.

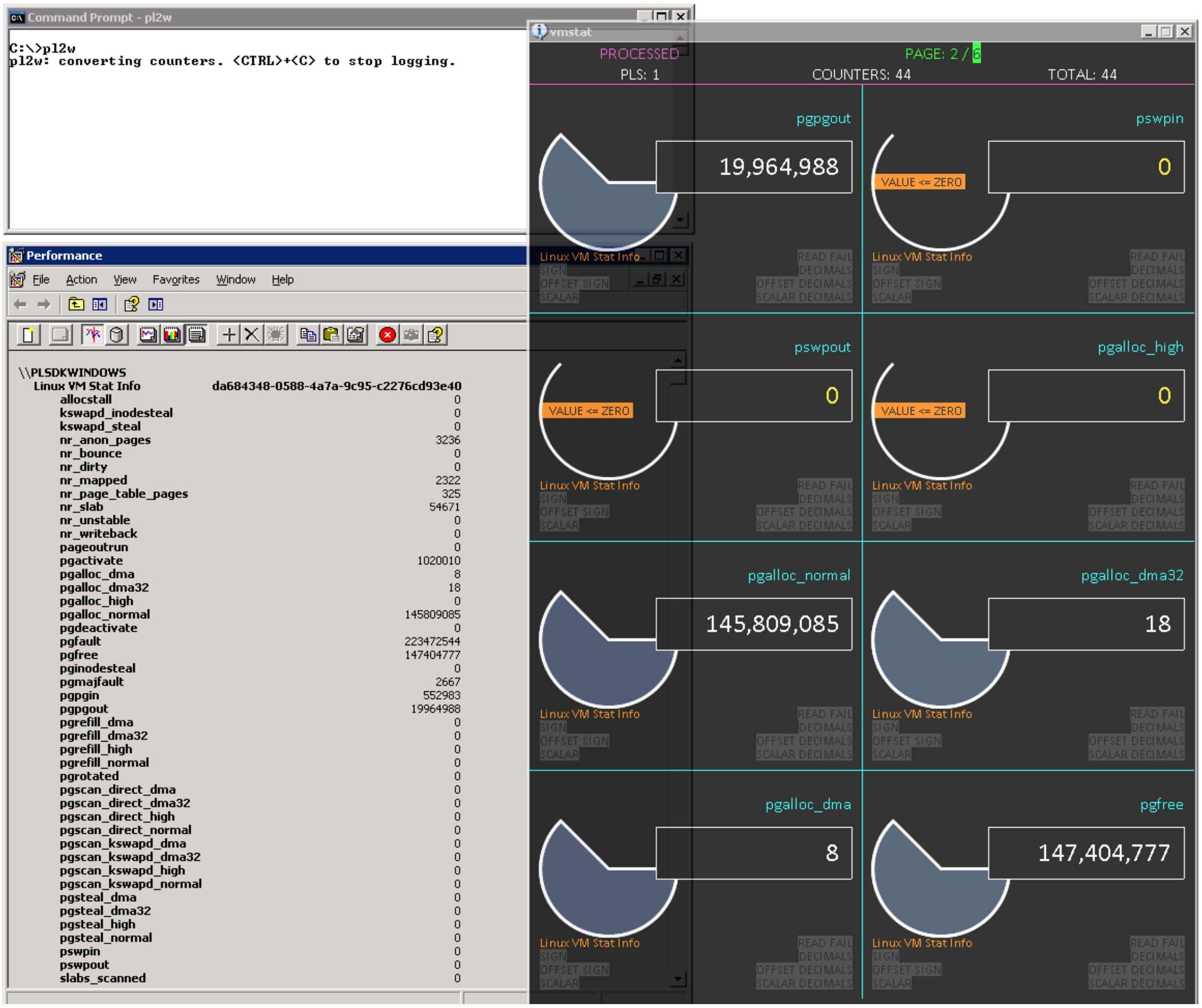
Q50 - Can I use the API if I do not have the source code of an application?

A50 - Yes. The SDK is shipped with a set of scripting tools to create, read and write counters from the command line (and therefore from scripts or batch files). Many applications maintain custom log files or offer proprietary APIs to access application-specific performance data. These log files or APIs may require special access rights, so they may not be accessible to broader management software. If you have access to these log files or APIs, you can write a helper application to read this proprietary data export counters through the Intel® EC SDK. Please read chapter six of the Intel® Energy Checker User Guide for more details on the scripting tools provided with the Intel® EC SDK.

Q51 - Can I re-use existing performance counters in my code with the Intel® EC SDK?

A51 - Yes, if you already have existing performance counters in your code, it’s a relatively simple process to call the Intel® EC API whenever your internal code updates its counters. To optimize performance, it may be desirable to create a metrics manager thread as shown in the threading example below.

You can even use the Intel® Energy Checker API to exchange counter data between platforms. For example, if your instrumented application runs on Linux*, it is possible to run the ESRV energy server on Solaris* 10 and capture the data on Windows* using the companion/sample PL CSV Logger tool. Since the Windows* operating system provides a proprietary performance monitoring framework via the registry, your application may already use it to expose some performance/work related counters. You may also want to use an existing monitoring tool to collect the energy efficiency counters’ data. In both cases, you can use the Windows* interoperability tools shipped with the SDK. These tools allow the dynamic conversion of any Intel® Energy Checker counter into native Windows* counter and vice-versa. For example, it is possible to collect/monitor data exported by a Linux* application or system on a Windows* system as shown in the figure below. Please read chapter seven of the Intel® Energy Checker SDK User Guide for details.



```
C:\>pl2w
pl2w: converting counters. <CTRL>+<C> to stop logging.
```

```
\\PLSDKWINDOWS
  Linux VM Stat Info           da684348-0588-4a7a-9c95-c2276cd93e40
      allocstall                                                   0
      kswapd_inodesteal                                            0
      kswapd_steal                                                 0
      nr_anon_pages                                             3236
      nr_bounce                                                    0
      nr_dirty                                                     0
      nr_mapped                                                 2322
      nr_page_table_pages                                        325
      nr_slab                                                  54671
      nr_unstable                                                  0
      nr_writeback                                                 0
      pageoutrun                                                   0
      pgactivate                                             1020010
      pgalloc_dma                                                  8
      pgalloc_dma32                                               18
      pgalloc_high                                                 0
      pgalloc_normal                                       145809085
      pgdeactivate                                                 0
      pgfault                                              223472544
      pgfree                                               147404777
      pginodesteal                                                 0
      pgmajfault                                                2667
      pgpgin                                                  552983
      pgpgout                                               19964988
      pgrefill_dma                                                 0
      pgrefill_dma32                                               0
      pgrefill_high                                                0
      pgrefill_normal                                              0
      pgrotated                                                    0
      pgscan_direct_dma                                            0
      pgscan_direct_dma32                                          0
      pgscan_direct_high                                           0
      pgscan_direct_normal                                         0
      pgscan_kswapd_dma                                            0
      pgscan_kswapd_dma32                                          0
      pgscan_kswapd_high                                           0
      pgscan_kswapd_normal                                         0
      pgsteal_dma                                                  0
      pgsteal_dma32                                                0
      pgsteal_high                                                 0
      pgsteal_normal                                               0
      pswpin                                                       0
      pswpout                                                      0
      slabs_scanned                                                0
```

Q52 - Can I collect counter data with VTune?

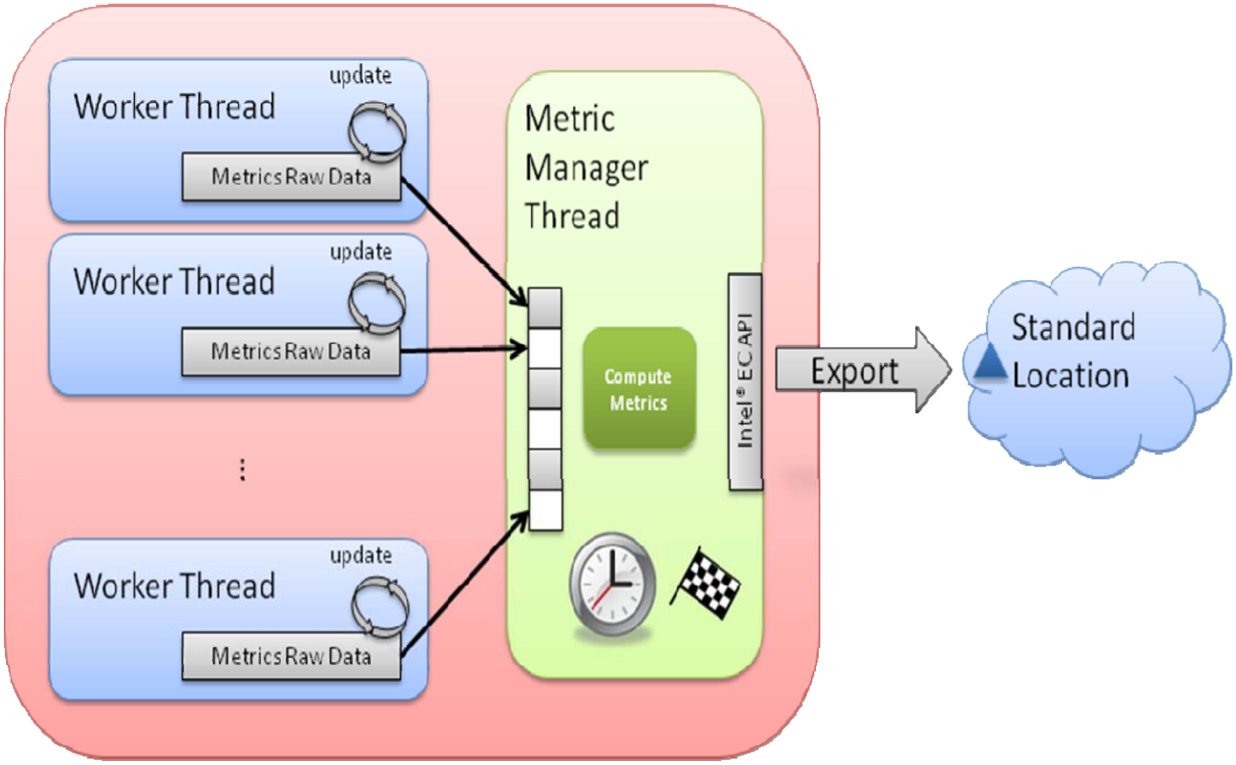
A52 - On a Windows* system, you can use the pl2w interoperability tool to convert any Intel® EC counter to a Windows* registry entry that can be read by the Intel® VTune™ Performance Analyzer like any other Windows* performance counter. Similarly, you can also collect these converted counters in Perfmon* or similar tools.

Q53 - Can I use the API in a threaded application?

A53 - Yes, the API is thread-safe.

Q54 - **What is the best instrumentation strategy for threaded code?**

A54 - As always, it depends on your application. However, to minimize the impact on your code, we recommend dedicating a thread to manage the data collection. With this model, your worker threads just need to account for the work they do – which is likely already implemented – and store their counters in shared variables that can be read by the metric manager thread. The dedicated metric manager thread can aggregate the work counts, optionally compute energy efficiency metric, and export the aggregated data and/or metrics at a pace appropriate to the application. Please read §9.7 of the Intel® Energy Checker User Guide for more details.



Note that the metric manager thread in the figure above can determine the frequency at which it exports counters through the Intel® EC API. Some counters may be updated at a different frequency than other counters.

Q55 - **What types of programs can use the Intel® EC SDK? Can I use the API from managed code?**

A55 - The Intel® Energy Checker SDK supports the following operating systems: Windows*, Linux*, Solaris* 10 and MacOS* X; the API transparently manages counter exchanges across these platforms. The API is written in C and can therefore be compiled with any standard C compiler and linked with objects generated by compilers for most compiled languages. The source code of the Intel® EC API provided in the Intel® EC SDK was successfully compiled with the Intel® C/C++ compiler 11.1.040, Microsoft Visual Studio 2005, GCC 4.1.2 on Linux, GCC 4.0.1 on MacOS X, and GCC 3.4.3 on Solaris 10. It should consequently work with any standard C/C++ compiler.

The API has a JNI* interface for Java*, and the SDK User Guide provides sample wrapping classes for Java and C#. A tool such as SWIG* can be used to interface with various scripting languages. The SDK also provides a set of scripting tools for UNIX*-like systems and MS-DOS* systems to provide the API's functionality from a script or batch file.

Q56 - How can I resolve undefined symbol errors when I try to build the API code?

A56 - There are certain mandatory symbols that need to be defined when building code with the Intel® Energy Checker API. For example, one of the mandatory symbols defines what OS you are building for, such as __PL_WINDOWS__ for Windows* or __PL_SOLARIS__ for Solaris* 10. Please read §3.9 of the Intel® Energy Checker SDK for details on which symbols you need to define when building the API code.

Q57 - Where is the counter data stored?

A57 - This is implementation-specific and should not impact the design of your application. Opening, read, writing, and closing a PL is handled independent of the storage method. In the initial release of the Intel® EC SDK (and for the foreseeable future), counter data is stored in files and is managed via the operating system's file system.

Q58 - Which network file system can I use to aggregate the data?

A58 - The Intel® Energy Checker is agnostic regarding the networked file system you are using to aggregate the data in a distributed environment. Since the API guarantees that you can aggregate all the counters without data collision [unless you explicitly re-use previous PL's], you can select the file system of your choice. We've chosen to illustrate the use of NFS* (Network File System) in the Intel® EC SDK User Guide examples since NFS is widely available, has acceptable performance for our use, and easily shares data between Windows* and UNIX* (with aggregation on a Windows* node).

Q59 - How can I erase a PL?

A59 - The current implementation of the Intel® EC SDK stores all counters within a PL in a folder whose name is based on the name of the application creating the PL and the PL ID returned to that application. Assuming you have the appropriate access rights and the PL is not used by any other application, simply delete the PL folder corresponding to that application name and ID. Refer to §4.1 of the Intel® EC SDK User Guide for more information on default locations for PL folders.

Q60 - If I want to reference the SDK, how should I do it?

A60 - The official name of the SDK is the **Intel® Energy Checker SDK**, which can be abbreviated as the **Intel® EC SDK**.

Q61 - What is the state of the product on the WhatIf.intel.com site?

A61 - The Intel® Energy Checker SDK is provided free of charge as a technology preview. We're interested in your feedback. Since it is only a technology preview, the availability and support for Intel® Energy Checker SDK is not long-term via the WhatIf site. Intel may release a derivative Intel® Energy Checker product in the future.

Q62 - How do I report problems or send feedback?

A62 - You are welcome to join our What If Software Forum to post your questions and issues. The team will keep an eye on the discussion and do our best to answer your questions.

Q63 - What kind of feedback are you looking for?

A63 - Please visit the What If Software Forum and provide your thoughts in an Intel® Energy Checker discussion thread or create a new one. We appreciate all feedback, and are particularly interested feedback on the following:

- What are your thoughts on this methodology for studying software energy efficiency?
- How does it compare with other techniques you have used?
- What features of the Intel® Energy Checker SDK do you like? What did you find difficult to use or to understand?
- What is the biggest obstacle that has stopped you from instrumenting your application for energy efficiency analysis? Does the Intel® Energy Checker address this obstacle?
- Did you find any bugs in the Intel® Energy Checker SDK? We are always striving to improve the quality of our product.
- Do you have an interesting example of the Intel® Energy Checker SDK has been useful to your organization?
- Are there any features you would like to see added to the Intel® Energy Checker SDK?

*Third-party trademarks are the property of their respective owners.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document 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.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/#/en_US_01

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, go to: http://www.intel.com/performance/resources/benchmark_limitations.htm

What If Pre-Release License Agreement:

IMPORTANT - READ BEFORE COPYING, INSTALLING OR USING.

Do not copy, install, or use the "Materials" provided under this license agreement ("Agreement"), until you have carefully read the following terms and conditions. By copying, installing, or otherwise using the Materials, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, do notcopy, install, or use the Materials.

Pre-Release License Agreement for Pre-Release Software

1. PRE-RELEASE: The Materials are pre-release code, which may not be fully functional and which Intel may substantially modify in producing any final version. Intel can provide no assurance that it will ever produce or make generally available a final version.

2. LICENSE DEFINITIONS:

A. "Materials" are defined as the software, documentation, license key codes and other materials, including any updates and upgrade thereto, for the applicable pre-release software (which may be found at http://whatif.intel.com/), that are provided to you under this Agreement.

3. LICENSE GRANT:

A. Subject to all of the terms and conditions of this Agreement, Intel Corporation ("Intel") grants to you a non-exclusive, non-assignable copyright license to make only the minimum number of copies of the Materials reasonably necessary for your internal testing and development of your products.

B. Subject to all of the terms and conditions of this Agreement, Intel grants to you a non-exclusive, non-assignable copyright license to modify the Materials that are provided in source code (human readable) form.

C. If the Materials include the file named "redist.txt", then subject to all of the terms and conditions of this Agreement and any specific restrictions which may appear in the "redist.txt" file, Intel grants to you a non-exclusive, non-assignable copyright license to redistribute the files (unmodified or modified by you) listed in the "redist.txt" file only as part of the application you develop with the Materials.

4. LICENSE RESTRICTIONS:

A. You may not reverse-assemble, reverse-compile, or otherwise reverse-engineer any software provided solely in binary form.

B. You may not distribute any portion of Materials, whether in source or binary form, to any third party, except as specified in this Agreement.

5. COPYRIGHT: Title to the Materials and all copies thereof remain with Intel or its suppliers. The Materials are copyrighted and are protected by United States copyright laws and international treaty provisions. You will not remove any copyright notice from the Materials. You agree to prevent any unauthorized copying of the Materials. Except as expressly provided herein, Intel does not grant any express or implied right to you under Intel patents, copyrights, trademarks, or trade secret information. Subject to Intel's ownership of the Materials, all right, title and interest in and to your modifications shall belong to you.

6. REPLACEMENTS: The Materials are provided "AS IS" without warranty of any kind. If the media on which the Materials are furnished are found to be defective in material or workmanship under normal use for a period of ninety (90) days from the date of receipt, Intel's entire liability and your exclusive remedy shall be the replacement of the media. This offer is void if the media defect results from accident, abuse, or misapplication.

7. LIMITATION OF LIABILITY: THE ABOVE REPLACEMENT PROVISION IS THE ONLY WARRANTY OF ANY KIND. INTEL OFFERS NO OTHER WARRANTY EITHER EXPRESS OR IMPLIED INCLUDING THOSE OF MERCHANTABILITY, NONINFRINGEMENT OF THIRD- PARTY INTELLECTUAL PROPERTY OR FITNESS FOR A PARTICULAR PURPOSE. NEITHER INTEL NOR ITS SUPPLIERS SHALL BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF INTEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

8. UNAUTHORIZED USE: THE MATERIALS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE IN ANY TYPE OF SYSTEM OR APPLICATION IN WHICH THE FAILURE OF THE MATERIALS COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR (E.G MEDICAL SYSTEMS, LIFE SUSTAINING OR LIFE SAVING SYSTEMS). Should the buyer purchase or use the Materials for any such unintended or unauthorized use, the buyer shall indemnify and hold Intel and its officers, subsidiaries and affiliates harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of product liability, personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Intel was negligent regarding the design or manufacture of the part.

9. USER SUBMISSIONS: You agree that any material, information or other communication, including all data, images, sounds, text, and other things embodied therein, you transmit or post to an Intel website or provide to Intel under this Agreement will be considered non-confidential ("Communications"). Intel will have no confidentiality obligations with respect to the Communications. You agree that Intel and its designees will be free to copy, modify, create derivative works, publicly display, disclose, distribute, license and sublicense through multiple tiers of distribution and licensees, incorporate and otherwise use the Communications, including derivative works thereto, for any and all commercial or non-commercial purposes.

10. TERMINATION OF THIS LICENSE: The term of this Agreement will commence on the date this Agreement is accepted by You and will continue until terminated. This Agreement will terminate without notice on the last day of the pre-release period, which is specified elsewhere in the Materials, or upon the commercial release of the Materials. Intel may terminate this Agreement at any time, with or without cause, with written notice to you. Upon termination, you will immediately destroy the Materials or return all copies of the Materials to Intel along with any copies you have made.

11. U.S. GOVERNMENT RESTRICTED RIGHTS: The Materials are provided with "RESTRICTED RIGHTS". Use, duplication or disclosure by the Government is subject to restrictions set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor. Use of the Materials by the Government constitutes acknowledgment of Intel's rights in them.

12. EXPORT RESTRICTED RIGHTS: This software is subject to the U.S. Export Administration Regulations and other U.S. law, and may not be exported or re-exported to certain countries (Burma, Cuba, Iran, North Korea, Sudan, and Syria) or to persons or entities prohibited from receiving U.S. exports (including Denied Parties, Specially Designated Nationals, and entities on the Bureau of Export Administration Entity List or involved with missile technology or nuclear, chemical or biological weapons).

13. APPLICABLE LAWS: Any claim arising under or relating to this Agreement shall be governed by the internal substantive laws of the State of Delaware or federal courts located in Delaware, without regard to principles of conflict of laws. You may not export the Materials in violation of applicable export laws.