msaSDK Module

`.utils.sysinfo`

Provides System Information about devices, OS etc.

Classes

MSACPUFrequency

Bases: `SQLModel`

Pydantic CPU Frequency Info Model.

Attributes

current `class-attribute`

```
current: Optional[float]
```

max `class-attribute`

```
max: Optional[int]
```

min `class-attribute`

```
min: Optional[int]
```

MSACPUStats

Bases: `SQLModel`

Pydantic CPU Stats Info Model.

Attributes

ctx_switches `class-attribute`

```
ctx_switches: Optional[int]
```

> number of context switches (voluntary + involuntary) since boot.

interrupts `class-attribute`

```
interrupts: Optional[int]
```

> number of interrupts since boot.

soft_interrupts `class-attribute`

```
soft_interrupts: Optional[int]
```

> number of software interrupts since boot. Always set to 0 on Windows and SunOS.

syscalls `class-attribute`

```
syscalls: Optional[int]
```

> number of system calls since boot. Always set to 0 on Linux.

MSACPUTimes

Bases: `SQLModel`

Pydantic CPU Timings Info Model.

Attributes

guest `class-attribute`

```
guest: Optional[float]
```

> (Linux 2.6.24+): time spent running a virtual CPU for guest operating systems under the control of the Linux kernel

guest_nice `class-attribute`

```
guest_nice: Optional[int]
```

> (Linux 3.2.0+): time spent running a niced guest (virtual CPU for guest operating systems under the control of the Linux kernel)

idle `class-attribute`

```
idle: Optional[float]
```

> time spent doing nothing

iowait `class-attribute`

```
iowait: Optional[float]
```

> (Linux): time spent waiting for I/O to complete. This is not accounted in idle time counter.

irq `class-attribute`

```
irq: Optional[int]
```

> (Linux, BSD): time spent for servicing hardware interrupts

nice `class-attribute`

```
nice: Optional[int]
```

> (UNIX): time spent by niced (prioritized) processes executing in user mode; on Linux this also includes guest_nice time

softirq `class-attribute`

```
softirq: Optional[float]
```

> (Linux): time spent for servicing software interrupts

steal `class-attribute`

```
steal: Optional[int]
```

> (Linux 2.6.11+): time spent by other operating systems running in a virtualized environment

system `class-attribute`

```
system: Optional[float]
```

> time spent by processes executing in kernel mode

user `class-attribute`

```
user: Optional[float]
```

> time spent by normal processes executing in user mode; on Linux this also includes guest time

MSADiskIO

Bases: `SQLModel`

Pydantic Disk IO Info Model.

ATTRIBUTE	DESCRIPTION
`read_count`	number of reads **TYPE:** `Optional[int]`
`write_count`	number of writes **TYPE:** `Optional[int]`
`read_bytes`	number of bytes read **TYPE:** `Optional[int]`
`write_bytes`	number of bytes written **TYPE:** `Optional[int]`
`read_time`	(all except NetBSD and OpenBSD) time spent reading from disk (in milliseconds) **TYPE:** `Optional[int]`
`write_time`	(all except NetBSD and OpenBSD) time spent writing to disk (in milliseconds) **TYPE:** `Optional[int]`
`busy_time`	(Linux, FreeBSD) time spent doing actual I/Os (in milliseconds) **TYPE:** `Optional[int]`
`read_merged_count`	number of merged reads (see iostats doc) **TYPE:** `Linux`
`write_merged_count`	number of merged writes (see iostats doc) **TYPE:** `Linux`

Attributes

busy_time `class-attribute`

```
busy_time: Optional[int]
```

read_bytes `class-attribute`

```
read_bytes: Optional[int]
```

read_count `class-attribute`

```
read_count: Optional[int]
```

read_merged_count `class-attribute`

```
read_merged_count: Optional[int]
```

read_time `class-attribute`

```
read_time: Optional[int]
```

write_bytes `class-attribute`

```
write_bytes: Optional[int]
```

write_count `class-attribute`

```
write_count: Optional[int]
```

write_merged_count `class-attribute`

```
write_merged_count: Optional[int]
```

write_time `class-attribute`

```
write_time: Optional[int]
```

MSAGPUInfo

Bases: `SQLModel`

Pydantic GPU Info Model.

Attributes

free_memory `class-attribute`

```
free_memory: Optional[str]
```

id `class-attribute`

```
id: Optional[int]
```

load `class-attribute`

```
load: Optional[str]
```

name `class-attribute`

```
name: Optional[str]
```

temperature `class-attribute`

```
temperature: Optional[str]
```

total_memory `class-attribute`

```
total_memory: Optional[str]
```

used_memory `class-attribute`

```
used_memory: Optional[str]
```

uuid `class-attribute`

```
uuid: Optional[str]
```

MSAMemoryUsage

Bases: `SQLModel`

Pydantic Memory Usage Info Model.

Attributes

active `class-attribute`

```
active: Optional[float]
```

> (UNIX): memory currently in use or very recently used, and so it is in RAM.

available `class-attribute`

```
available: Optional[float]
```

> the memory that can be given instantly to processes without the system going into swap. This is calculated by summing different memory values depending on the platform and it is supposed to be used to monitor actual memory usage in a cross platform fashion.

buffers `class-attribute`

```
buffers: Optional[float]
```

> (Linux, BSD): cache for things like file system metadata.

cached `class-attribute`

```
cached: Optional[float]
```

> (Linux, BSD): cache for various things.

free `class-attribute`

```
free: Optional[float]
```

> memory not being used at all (zeroed) that is readily available; note that this doesn't reflect the actual memory available (use available instead). total - used does not necessarily match free.

inactive `class-attribute`

```
inactive: Optional[float]
```

> (UNIX): memory that is marked as not used.

percent `class-attribute`

```
percent: Optional[float]
```

> the percentage usage calculated as (total - available) / total * 100

total `class-attribute`

```
total: Optional[float]
```

> total physical memory (exclusive swap).

used `class-attribute`

```
used: Optional[float]
```

> memory used, calculated differently depending on the platform and designed for informational purposes only. total - free does not necessarily match used.

MSANetworkAdapter

Bases: `SQLModel`

Pydantic Network Adapter Info Model.

Attributes

address `class-attribute`

```
address: Optional[str]
```

> the primary NIC address (always set).

broadcast `class-attribute`

```
broadcast: Optional[str]
```

> the broadcast address (may be None).

family `class-attribute`

```
family: Optional[int]
```

> the address family, either AF_INET or AF_INET6 or psutil.AF_LINK, which refers to a MAC address.

netmask `class-attribute`

```
netmask: Optional[str]
```

> the netmask address (may be None).

ptp `class-attribute`

```
ptp: Optional[int]
```

> stands for "point to point"; it's the destination address on a point to point interface (typically a VPN). broadcast and ptp are mutually exclusive. May be None.

MSANetworkAdapters

Bases: `SQLModel`

Pydantic Network Adapters List Model.

Attributes

adapters `class-attribute`

```
adapters: List[MSANetworkAdapter] = []
```

name `class-attribute`

```
name: str = ''
```

MSANetworkConnection

Bases: `SQLModel`

Pydantic Network Connection Info Model.

Attributes

family `class-attribute`

```
family: Optional[int]
```

> the address family, either AF_INET, AF_INET6 or AF_UNIX.

file_descriptor `class-attribute`

```
file_descriptor: Optional[int]
```

> the socket file descriptor. If the connection refers to the current process this may be passed to socket.fromfd to obtain a usable socket object. On Windows and SunOS this is always set to -1.

index `class-attribute`

```
index: Optional[int]
```

local_addr `class-attribute`

```
local_addr: Optional[str]
```

> the local address as a `(ip, port)` named tuple or a `path` in case of AF_UNIX sockets. For UNIX sockets see notes below.

pid `class-attribute`

```
pid: Optional[int]
```

> the PID of the process which opened the socket, if retrievable, else `None`. On some platforms (e.g. Linux) the availability of this field changes depending on process privileges (root is needed).

remote_addr `class-attribute`

```
remote_addr: Optional[str]
```

> the remote address as a `(ip, port)` named tuple or an absolute `path` in case of UNIX sockets. When the remote endpoint is not connected you'll get an empty tuple (AF_INET*) or `""` (AF_UNIX). For UNIX sockets see notes below.

status `class-attribute`

```
status: str = ''
```

> represents the status of a TCP connection. The return value is one of the `psutil.CONN_*` constants (a string). For UDP and UNIX sockets this is always going to be psutil.CONN_NONE.

type `class-attribute`

```
type: Optional[int]
```

> the address type, either `SOCK_STREAM`, `SOCK_DGRAM` or `SOCK_SEQPACKET`.

MSANetworkIO

Bases: `SQLModel`

Pydantic Network IO Info Model.

ATTRIBUTE	DESCRIPTION
`bytes_sent`	number of bytes sent **TYPE:** `Optional[int]`
`bytes_recv`	number of bytes received **TYPE:** `Optional[int]`
`packets_sent`	number of packets sent **TYPE:** `Optional[int]`
`packets_recv`	number of packets received **TYPE:** `Optional[int]`
`errin`	total number of errors while receiving **TYPE:** `Optional[int]`
`errout`	total number of errors while sending **TYPE:** `Optional[int]`
`dropin`	total number of incoming packets which were dropped **TYPE:** `Optional[int]`
`dropout`	total number of outgoing packets which were dropped (always 0 on macOS and BSD) **TYPE:** `Optional[int]`

Attributes

bytes_recv `class-attribute`

```
bytes_recv: Optional[int]
```

bytes_sent `class-attribute`

```
bytes_sent: Optional[int]
```

dropin `class-attribute`

```
dropin: Optional[int]
```

dropout `class-attribute`

```
dropout: Optional[int]
```

errin `class-attribute`

```
errin: Optional[int]
```

errout `class-attribute`

```
errout: Optional[int]
```

packets_recv `class-attribute`

```
packets_recv: Optional[int]
```

packets_sent `class-attribute`

```
packets_sent: Optional[int]
```

MSANetworkStat

Bases: `SQLModel`

Pydantic Network Stats Info Model.

Attributes

duplex `class-attribute`

```
duplex: Optional[int]
```

> the duplex communication type; it can be either NIC_DUPLEX_FULL, NIC_DUPLEX_HALF or NIC_DUPLEX_UNKNOWN.

isup `class-attribute`

```
isup: Optional[bool]
```

> a bool indicating whether the NIC is up and running (meaning ethernet cable or Wi-Fi is connected).

mtu `class-attribute`

```
mtu: Optional[int]
```

> NIC's maximum transmission unit expressed in bytes.

speed `class-attribute`

```
speed: Optional[int]
```

> the NIC speed expressed in mega bits (MB), if it can't be determined (e.g. 'localhost') it will be set to 0.

MSANetworkStats

Bases: `SQLModel`

Pydantic Network Stats List Info Model.

Attributes

adapters `class-attribute`

```
adapters: List[MSANetworkStat] = []
```

name `class-attribute`

```
name: str = ''
```

MSASwap

Bases: `SQLModel`

Pydantic Swapfile Info Model.

Attributes

free `class-attribute`

```
free: Optional[float]
```

percent `class-attribute`

```
percent: Optional[float]
```

> the percentage usage calculated as (total - available) / total * 100

total `class-attribute`

```
total: Optional[float]
```

used `class-attribute`

```
used: Optional[float]
```

MSASystemGPUInfo

Bases: `SQLModel`

Pydantic System GPU Info Model.

Attributes

CPU_Logical `class-attribute`

```
CPU_Logical: Optional[int]
```

CPU_Physical `class-attribute`

```
CPU_Physical: Optional[int]
```

GPUs `class-attribute`

```
GPUs: Optional[List[MSAGPUInfo]]
```

HW_Identifier `class-attribute`

```
HW_Identifier: str = ''
```

Host_Name `class-attribute`

```
Host_Name: str = ''
```

IP_Address `class-attribute`

```
IP_Address: str = ''
```

MAC_Address `class-attribute`

```
MAC_Address: str = ''
```

Memory_Available `class-attribute`

```
Memory_Available: str = ''
```

Memory_Physical `class-attribute`

```
Memory_Physical: str = ''
```

Node_Name `class-attribute`

```
Node_Name: str = ''
```

OS_Name `class-attribute`

```
OS_Name: str = ''
```

OS_Release `class-attribute`

```
OS_Release: str = ''
```

OS_Version `class-attribute`

```
OS_Version: str = ''
```

PID `class-attribute`

```
PID: Optional[int]
```

Runtime_Cmd `class-attribute`

```
Runtime_Cmd: List[str] = []
```

Runtime_Exe `class-attribute`

```
Runtime_Exe: str = ''
```

Runtime_Status `class-attribute`

```
Runtime_Status: str = ''
```

Service_Start `class-attribute`

```
Service_Start: str = ''
```

System_Boot `class-attribute`

```
System_Boot: str = ''
```

MSASystemInfo

Bases: `SQLModel`

Pydantic System Info Model.

Attributes

CPU_Affinity `class-attribute`

```
CPU_Affinity: Optional[int]
```

CPU_Current `class-attribute`

```
CPU_Current: Optional[int]
```

CPU_Frequency `class-attribute`

```
CPU_Frequency: Optional[MSACPUFrequency]
```

CPU_LoadAvg `class-attribute`

```
CPU_LoadAvg: Optional[List[float]]
```

CPU_Logical `class-attribute`

```
CPU_Logical: Optional[int]
```

> Amount of logical (each physical core doing 2 or more threads, hyperthreading) CPU's

CPU_Physical `class-attribute`

```
CPU_Physical: Optional[int]
```

> Amount of physical CPU's

CPU_Stats `class-attribute`

```
CPU_Stats: Optional[MSACPUStats]
```

CPU_Times `class-attribute`

```
CPU_Times: Optional[MSACPUTimes]
```

CPU_Usage_Name `class-attribute`

```
CPU_Usage_Name: str = ''
```

CPU_Usage_Process `class-attribute`

```
CPU_Usage_Process: Optional[float]
```

CPU_Usage_Total `class-attribute`

```
CPU_Usage_Total: Optional[float]
```

Disk_IO `class-attribute`

```
Disk_IO: Optional[MSADiskIO]
```

HW_Identifier `class-attribute`

```
HW_Identifier: str = ''
```

Host_Name `class-attribute`

```
Host_Name: str = ''
```

IP_Address `class-attribute`

```
IP_Address: str = ''
```

MAC_Address `class-attribute`

```
MAC_Address: str = ''
```

Memory_Available `class-attribute`

```
Memory_Available: str = ''
```

Memory_Physical `class-attribute`

```
Memory_Physical: str = ''
```

Memory_Usage `class-attribute`

```
Memory_Usage: Optional[MSAMemoryUsage]
```

Network_Adapters `class-attribute`

```
Network_Adapters: Optional[List[MSANetworkAdapters]]
```

Network_Connections `class-attribute`

```
Network_Connections: Optional[List[MSANetworkConnection]]
```

Network_IO `class-attribute`

```
Network_IO: Optional[MSANetworkIO]
```

Network_Stats `class-attribute`

```
Network_Stats: Optional[List[MSANetworkStats]]
```

Node_Name `class-attribute`

```
Node_Name: str = ''
```

OS_Name `class-attribute`

```
OS_Name: str = ''
```

OS_Release `class-attribute`

```
OS_Release: str = ''
```

OS_Version `class-attribute`

```
OS_Version: str = ''
```

PID `class-attribute`

```
PID: Optional[int]
```

Runtime_Cmd `class-attribute`

```
Runtime_Cmd: List[str] = []
```

Runtime_Exe `class-attribute`

```
Runtime_Exe: str = ''
```

Runtime_Status `class-attribute`

```
Runtime_Status: str = ''
```

> Service Status, running or stopped

Service_Start `class-attribute`

```
Service_Start: str = ''
```

Swap `class-attribute`

```
Swap: Optional[MSASwap]
```

System_Boot `class-attribute`

```
System_Boot: str = ''
```

Temperatures `class-attribute`

```
Temperatures: Optional[List[MSATemperatures]]
```

MSATemperature

Bases: `SQLModel`

Pydantic Temperature Info Model.

Attributes

critical `class-attribute`

```
critical: Optional[float]
```

current `class-attribute`

```
current: Optional[float]
```

high `class-attribute`

```
high: Optional[float]
```

label `class-attribute`

```
label: Optional[str]
```

MSATemperatures

Bases: `SQLModel`

Pydantic Temperatures List Model.

Attributes

device `class-attribute`

```
device: str = ''
```

temps `class-attribute`

```
temps: List[MSATemperature] = []
```

Functions

get_cpu_freq

```
get_cpu_freq() -> MSACPUFrequency
```

Get psutil.cpu_freq()

RETURNS	DESCRIPTION
`cpf`	MSACPUFrequency **TYPE:** `MSACPUFrequency`

get_cpu_stats

```
get_cpu_stats() -> MSACPUStats
```

Get psutil.cpu_times()

RETURNS	DESCRIPTION

RETURNS	DESCRIPTION
`cst`	MSACPUStats **TYPE:** `MSACPUStats`

get_cpu_times

```
get_cpu_times() -> MSACPUTimes
```

Get psutil.cpu_times()

RETURNS	DESCRIPTION
`cti`	MSACPUTimes **TYPE:** `MSACPUTimes`

get_cpu_usage

```
get_cpu_usage(
    user: str = None, ignore_self: bool = False
) -> tuple[int, int, str]
```

Returns the total CPU usage for all available cores.

PARAMETER	DESCRIPTION
`user`	If given, returns only the total CPU usage of all processes for the given user. **TYPE:** `str` **DEFAULT:** `None`
`ignore_self`	If `True` the process that runs this script will be ignored. **TYPE:** `bool` **DEFAULT:** `False`

RETURNS	DESCRIPTION
`total`	total usage **TYPE:** `int`

RETURNS	DESCRIPTION
`largest_process`	largest process usage **TYPE:** `int`
`largest_process_name`	name of the largest process **TYPE:** `str`

get_disk_io

```
get_disk_io() -> MSADiskIO
```

Get psutil.disk_io_counters()

RETURNS	DESCRIPTION
`dio`	MSADiskIO **TYPE:** `MSADiskIO`

get_gpus

```
get_gpus() -> List[MSAGPUInfo]
```

Get GPUtil.getGPUs()

RETURNS	DESCRIPTION
`list_gpus`	List[MSAGPUInfo] = [] **TYPE:** `List[MSAGPUInfo]`

get_hostname

```
get_hostname() -> str
```

Get socket.gethostname()

RETURNS	DESCRIPTION

RETURNS	DESCRIPTION
`hostname`	str **TYPE:** `str`

get_list_partitions

```
get_list_partitions() -> List
```

Get psutil.disk_partitions()

RETURNS	DESCRIPTION
`partitions_list`	List = [] **TYPE:** `List`

get_load_average

```
get_load_average() -> tuple[float, float, float]
```

Returns the CPU load average in tuple[1min, 5min, 15min].

RETURNS	DESCRIPTION
`1min`	total usage **TYPE:** `float`
`5min`	largest process usage **TYPE:** `float`
`15min`	name of the largest process **TYPE:** `float`

get_map_disk_usage

```
get_map_disk_usage() -> Dict
```

Get get_partition_usage(get_list_partitions())

RETURNS	DESCRIPTION
`rdict`	Dict **TYPE:** `Dict`

get_memory_usage

```
get_memory_usage() -> MSAMemoryUsage
```

Get psutil.virtual_memory()

RETURNS	DESCRIPTION
`mu`	MSAMemoryUsage **TYPE:** `MSAMemoryUsage`

get_network_adapters

```
get_network_adapters() -> List[MSANetworkAdapters]
```

Get psutil.net_if_addrs()

RETURNS	DESCRIPTION
`ret`	List[MSANetworkAdapters] = [] **TYPE:** `List[MSANetworkAdapters]`

get_network_connections

```
get_network_connections() -> List[MSANetworkConnection]
```

Get psutil.net_connections()

RETURNS	DESCRIPTION
`rlist`	List[MSANetworkConnection] = [] **TYPE:** `List[MSANetworkConnection]`

get_network_io

```
get_network_io() -> MSANetworkIO
```

> Get psutil.net_io_counters()
>
> | RETURNS | DESCRIPTION |
> |---|---|
> | `nio` | MSANetworkIO
TYPE: `MSANetworkIO` |

get_network_stats

```
get_network_stats() -> List[MSANetworkStats]
```

> Get psutil.net_if_stats()
>
> | RETURNS | DESCRIPTION |
> |---|---|
> | `ret` | List[MSANetworkStats] = []
TYPE: `List[MSANetworkStats]` |

get_partition_usage

```
get_partition_usage(partitions) -> Dict
```

> Get psutil.disk_usage(partition)
>
> | RETURNS | DESCRIPTION |
> |---|---|
> | `ret` | Dict = {"partition": list, "total": list, "used": list, "free": list, "percent": list}
TYPE: `Dict` |

get_swap

```
get_swap() -> MSASwap
```

> Get psutil.swap_memory()

RETURNS	DESCRIPTION
`sw`	MSASwap **TYPE:** `MSASwap`

get_sysgpuinfo

```
get_sysgpuinfo() -> MSASystemGPUInfo
```

Get MSASystemGPUInfo

RETURNS	DESCRIPTION
`system_gpu_info`	Pydantic System GPU Info Model. **TYPE:** `MSASystemGPUInfo`

get_sysinfo

```
get_sysinfo() -> MSASystemInfo
```

Get MSASystemInfo

RETURNS	DESCRIPTION
`system_info`	Pydantic System Info Model. **TYPE:** `MSASystemInfo`

get_temperatures

```
get_temperatures() -> List[MSATemperatures]
```

Get psutil.sensors_temperatures()

RETURNS	DESCRIPTION
`ret`	List[MSATemperatures] = [] **TYPE:** `List[MSATemperatures]`

Last update: September 14, 2022

Created: September 14, 2022