1 Zabbix agent

Overview

These checks use the communication with Zabbix agent for data gathering.

There are passive and active agent checks. When configuring an item, you can select the required type:

- *Zabbix agent* - for passive checks
- *Zabbix agent (active)* - for active checks

Supported item keys

The table provides details on the item keys that you can use with Zabbix agent items.

See also:

- Items supported by platform
- Item keys specific for Windows agent

Mandatory and optional parameters

Parameters without angle brackets are mandatory. Parameters marked with angle brackets **< >** are optional.

Key			
Description	**Return value**	**Parameters**	**Comments**
agent.hostname			
Agent host name.	String		Returns the actual value of the agent hostname from a configuration file.
agent.ping			
Agent availability check.	Nothing - unavailable 1 - available		Use the **nodata()** trigger function to check for host unavailability.
agent.version			
Version of Zabbix agent.	String		Example of returned value: 1.8.2
kernel.maxfiles			
Maximum number of opened files supported by OS.	Integer		
kernel.maxproc			
Maximum number of processes supported by OS.	Integer		
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]			

Key			
Description	**Return value**	**Parameters**	**Comments**
Log file monitoring.	Log	**file** - full path and name of log file **regexp** - regular expression describing the required pattern **encoding** - code page identifier **maxlines** - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in zabbix_agentd.conf **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched text while an **\N** (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).	The item must be configured as an active check. If file is missing or permissions do not allow access, item turns unsupported. If `output` is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the `output` parameter is ignored. Content extraction using the `output` parameter takes place on the agent. Examples: ⇒ log[/var/log/syslog] ⇒ log[/var/log/syslog,error] ⇒ log[/home/zabbix/logs/logfile,,,100] *Example of using `output` parameter for extracting a number from log record:* log[/app1/app.log,"task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors",,,,\1]→ will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send only number 6080 to server. Because a number is being sent, the "Type of information" for this log item can be changed from "Log" to "Numeric (unsigned)" and the value can be used in graphs, triggers etc. *Example of using `output` parameter for rewriting log record before sending to server:* log[/app1/app.log,"([0-9 :-]+) task run ([0-9.]+) sec, processed ([0-9]+) records, ([0-9]+) errors",,,,"\1 RECORDS: \3, ERRORS: \4, DURATION: \2"]→ will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors " and send a modified record "2015-11-13 10:08:26 RECORDS: 6080, ERRORS: 0, DURATION: 6.08" to server. The `output` parameter is supported since Zabbix 2.2. The `mode` parameter is supported since Zabbix 2.0. See also additional information on log monitoring.
logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>]			

Key			
Description	**Return value**	**Parameters**	**Comments**
Log file monitoring with log rotation support.	Log	**file_regexp** - absolute path to file and regexp describing the file name pattern **regexp** - regular expression describing the required content pattern **encoding** - code page identifier **maxlines** - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in zabbix_agentd.conf **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched text while an **\N** (where N=1…9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).	The item must be configured as an active check. Log rotation is based on the last modification time of files. If `output` is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the `output` parameter is ignored. Content extraction using the `output` parameter takes place on the agent. Examples: ⇒ logrt["/home/zabbix/logs/^logfile[0-9]{1,3}$",,,100] → will match a file like "logfile1" (will not match ".logfile1") ⇒ logrt["/home/user/^logfile_.*_[0-9]{1,3}$","pattern_to_match","UTF-8",100] → will collect data from files such "logfile_abc_1" or "logfile__001". *Example of using `output` parameter for extracting a number from log record:* logrt[/app1/^test.*log$,"task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors",,,,\1]→ will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send only number 6080 to server. Because a number is being sent, the "Type of information" for this log item can be changed from "Log" to "Numeric (unsigned)" and the value can be used in graphs, triggers etc. *Example of using `output` parameter for rewriting log record before sending to server:* logrt[/app1/^test.*log$,"([0-9 :-]+) task run ([0-9.]+) sec, processed ([0-9]+) records, ([0-9]+) errors",,,,"\1 RECORDS: \3, ERRORS: \4, DURATION: \2"]→ will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors " and send a modified record "2015-11-13 10:08:26 RECORDS: 6080, ERRORS: 0, DURATION: 6.08" to server. The `output` parameter is supported since Zabbix 2.2. The `mode` parameter is supported since Zabbix 2.0. See also additional information on log monitoring.
net.dns[<ip>,name,<type>,<timeout>,<count>,<protocol>]			
Checks if DNS service is up.	0 - DNS is down (server did not respond or DNS resolution failed) 1 - DNS is up	**ip** - IP address of DNS server (leave empty for the default DNS server, ignored on Windows) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows) - number of tries for the request (default is 2) **protocol** - the protocol used to perform DNS queries: *udp* (default) or *tcp*	Example: ⇒ net.dns[8.8.8.8,zabbix.com,MX,2,1] The possible values for `type` are: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (except for Windows), *HINFO*, *MINFO*, *TXT*, *SRV* Internationalized domain names are not supported, please use IDNA encoded names instead. The `protocol` parameter is supported since Zabbix 3.0. SRV record type is supported since Zabbix agent versions 1.8.6 (Unix) and 2.0.0 (Windows). Naming before Zabbix 2.0 (still supported): *net.tcp.dns*
net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]			
Performs a DNS query.	Character string with the required type of information	**ip** - IP address of DNS server (leave empty for the default DNS server, ignored on Windows) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows) - number of tries for the request (default is 2) **protocol** - the protocol used to perform DNS queries: *udp* (default) or *tcp*	Example: ⇒ net.dns.record[8.8.8.8,zabbix.com,MX,2,1] The possible values for `type` are: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (except for Windows), *HINFO*, *MINFO*, *TXT*, *SRV* Internationalized domain names are not supported, please use IDNA encoded names instead. The `protocol` parameter is supported since Zabbix 3.0. SRV record type is supported since Zabbix agent versions 1.8.6 (Unix) and 2.0.0 (Windows). Naming before Zabbix 2.0 (still supported): *net.tcp.dns.query*
net.if.collisions[if]			

Key			
Description	**Return value**	**Parameters**	**Comments**
Number of out-of-window collisions.	Integer	**if** - network interface name	
net.if.discovery			
List of network interfaces. Used for low-level discovery.	JSON object		Supported since Zabbix agent version 2.0. On FreeBSD, OpenBSD and NetBSD supported since Zabbix agent version 2.2. Some Windows versions (for example, Server 2008) might require the latest updates installed to support non-ASCII characters in interface names.
net.if.in[if,<mode>]			
Incoming traffic statistics on network interface.	Integer	**if** - network interface name (Unix); network interface full description or IPv4 address (Windows) **mode** - possible values: *bytes* - number of bytes (default) *packets* - number of packets *errors* - number of errors *dropped* - number of dropped packets	On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. Multi-byte interface names on Windows are supported since Zabbix agent version 1.8.6. Examples: ⇒ net.if.in[eth0,errors] ⇒ net.if.in[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. You may use this key with a *Delta (speed per second)* store value in order to get bytes per second statistics.
net.if.out[if,<mode>]			
Outgoing traffic statistics on network interface.	Integer	**if** - network interface name (Unix); network interface full description or IPv4 address (Windows) **mode** - possible values: *bytes* - number of bytes (default) *packets* - number of packets *errors* - number of errors *dropped* - number of dropped packets	On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. Multi-byte interface names on Windows are supported since Zabbix agent 1.8.6 version. Examples: ⇒ net.if.out[eth0,errors] ⇒ net.if.out[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. You may use this key with a *Delta (speed per second)* store value in order to get bytes per second statistics.
net.if.total[if,<mode>]			
Sum of incoming and outgoing traffic statistics on network interface.	Integer	**if** - network interface name (Unix); network interface full description or IPv4 address (Windows) **mode** - possible values: *bytes* - number of bytes (default) *packets* - number of packets *errors* - number of errors *dropped* - number of dropped packets	On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. Examples: ⇒ net.if.total[eth0,errors] ⇒ net.if.total[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. You may use this key with a *Delta (speed per second)* store value in order to get bytes per second statistics. Note that dropped packets are supported only if both net.if.in and net.if.out work for dropped packets on your platform.
net.tcp.listen[port]			

Key			
Description	**Return value**	**Parameters**	**Comments**
Checks if this TCP port is in LISTEN state.	0 - it is not in LISTEN state 1 - it is in LISTEN state	**port** - TCP port number	Example: ⇒ net.tcp.listen[80] On Linux supported since Zabbix agent version 1.8.4 Since Zabbix 3.0.0, on Linux kernels 2.6.14 and above, information about listening TCP sockets is obtained from the kernel's NETLINK interface, if possible. Otherwise, the information is retrieved from /proc/net/tcp and /proc/net/tcp6 files.
net.tcp.port[<ip>,port]			
Checks if it is possible to make TCP connection to specified port.	0 - cannot connect 1 - can connect	**ip** - IP address (default is 127.0.0.1) **port** - port number	Example: ⇒ net.tcp.port[,80] → can be used to test availability of web server running on port 80. For simple TCP performance testing use net.tcp.service.perf[tcp,<ip>,<port>] Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually). Old naming: *check_port[*]*
net.tcp.service[service,<ip>,<port>]			
Checks if service is running and accepting TCP connections.	0 - service is down 1 - service is running	**service** - either of: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: ⇒ net.tcp.service[ftp,,45] → can be used to test the availability of FTP server on TCP port 45. Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually). Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use net.tcp.port for checks like these. Checking of LDAP and HTTPS by Windows agent is currently not supported. Note that the telnet check looks for a login prompt (':' at the end). See also known issues of checking HTTPS service. *https* and *telnet* services are supported since Zabbix 2.0. Old naming: *check_service[*]*
net.tcp.service.perf[service,<ip>,<port>]			
Checks performance of TCP service.	0 - service is down seconds - the number of seconds spent while connecting to the service	**service** - either of: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: ⇒ net.tcp.service.perf[ssh] → can be used to test the speed of initial response from SSH server. Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use net.tcp.service.perf[tcp,<ip>,<port>] for checks like these. Checking of LDAP and HTTPS by Windows agent is currently not supported. Note that the telnet check looks for a login prompt (':' at the end). See also known issues of checking HTTPS service. *https* and *telnet* services are supported since Zabbix 2.0. Old naming: *check_service_perf[*]*
net.udp.listen[port]			
Checks if this UDP port is in LISTEN state.	0 - it is not in LISTEN state 1 - it is in LISTEN state	**port** - UDP port number	Example: ⇒ net.udp.listen[68] On Linux supported since Zabbix agent version 1.8.4
net.udp.service[service,<ip>,<port>]			

Key			
Description	**Return value**	**Parameters**	**Comments**
Checks if service is running and responding to UDP requests.	0 - service is down 1 - service is running	**service** - *ntp* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: ⇒ net.udp.service[ntp,,45] → can be used to test the availability of NTP service on UDP port 45. This item is supported since Zabbix 3.0.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
net.udp.service.perf[service,<ip>,<port>]			
Checks performance of UDP service.	0 - service is down seconds - the number of seconds spent waiting for response from the service	**service** - *ntp* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: ⇒ net.udp.service.perf[ntp] → can be used to test response time from NTP service. This item is supported since Zabbix 3.0.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]			
Process CPU utilisation percentage.	Float	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **type** - CPU utilisation type: *total* (default), *user*, *system* **cmdline** - filter by command line (it is a regular expression) **mode** - data gathering mode: *avg1* (default), *avg5*, *avg15* **zone** - target zone: *current* (default), *all*. This parameter is supported only on Solaris platform. Since Zabbix 3.0.3 if agent has been compiled on Solaris without zone support but is running on a newer Solaris where zones are supported and <zone> parameter is default or *current* then the agent will return NOTSUPPORTED (the agent cannot limit results to only current zone). However, <zone> parameter value *all* is supported in this case.	Examples: ⇒ proc.cpu.util[,root] → CPU utilisation of all processes running under the "root" user ⇒ proc.cpu.util[zabbix_server,zabbix] → CPU utilisation of all zabbix_server processes running under the zabbix user The returned value is based on single CPU core utilisation percentage. For example CPU utilisation of a process fully using two cores is 200%. The process CPU utilisation data is gathered by a collector which supports the maximum of 1024 unique (by name, user and command line) queries. Queries not accessed during the last 24 hours are removed from the collector. This key is supported since Zabbix 3.0.0 and is available on several platforms (see Items supported by platform).
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]			
Memory used by process in bytes.	Integer	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **mode** - possible values: *avg*, *max*, *min*, *sum* (default) **cmdline** - filter by command line (it is a regular expression) **memtype** - type of memory used by process	Examples: ⇒ proc.mem[,root] → memory used by all processes running under the "root" user ⇒ proc.mem[zabbix_server,zabbix] → memory used by all zabbix_server processes running under the zabbix user ⇒ proc.mem[,oracle,max,oracleZABBIX] → memory used by the most memory-hungry process running under oracle having oracleZABBIX in its command line *Note*: When several processes use shared memory, the sum of memory used by processes may result in large, unrealistic values. See notes on selecting processes with `name` and `cmdline` parameters (Linux-specific). The `memtype` parameter is supported on several platforms since Zabbix 3.0.0.
proc.num[<name>,<user>,<state>,<cmdline>]			

Key			
Description	**Return value**	**Parameters**	**Comments**
The number of processes.	Integer	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **state** - possible values: *all* (default), *run*, *sleep*, *zomb* **cmdline** - filter by command line (it is a regular expression)	Examples: ⇒ proc.num[,mysql] → number of processes running under the mysql user ⇒ proc.num[apache2,www-data] → number of apache2 processes running under the www-data user ⇒ proc.num[,oracle,sleep,oracleZABBIX] → number of processes in sleep state running under oracle having oracleZABBIX in its command line See notes on selecting processes with `name` and `cmdline` parameters (Linux-specific). On Windows, only the `name` and `user` parameters are supported.
sensor[device,sensor,<mode>]			
Hardware sensor reading.	Float	**device** - device name **sensor** - sensor name **mode** - possible values: *avg*, *max*, *min* (if this parameter is omitted, device and sensor are treated verbatim).	Reads /proc/sys/dev/sensors on Linux 2.4. Example: ⇒ sensor[w83781d-i2c-0-2d,temp1] Prior to Zabbix 1.8.4, the *sensor[temp1]* format was used.
			Reads /sys/class/hwmon on Linux 2.6+. See a more detailed description of sensor item on Linux.
			Reads the *hw.sensors* MIB on OpenBSD. Examples: ⇒ sensor[cpu0,temp0] → temperature of one CPU ⇒ sensor["cpu[0-2]$",temp,avg] → average temperature of the first three CPU's Supported on OpenBSD since Zabbix 1.8.4.
system.boottime			
System boot time.	Integer (Unix timestamp)		
system.cpu.discovery			
List of detected CPUs/CPU cores. Used for low-level discovery.	JSON object		Supported on all platforms since 2.4.0.
system.cpu.intr			
Device interrupts.	Integer		
system.cpu.load[<cpu>,<mode>]			
CPU load.	Float	**cpu** - possible values: *all* (default), *percpu* (total load divided by online CPU count) **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*	Example: ⇒ system.cpu.load[,avg5] *percpu* is supported since Zabbix 2.0.0. Old naming: *system.cpu.loadX*
system.cpu.num[<type>]			
Number of CPUs.	Integer	**type** - possible values: *online* (default), *max*	Example: ⇒ system.cpu.num
system.cpu.switches			
Count of context switches.	Integer		Old naming: *system[switches]*
system.cpu.util[<cpu>,<type>,<mode>]			

Key			
Description	**Return value**	**Parameters**	**Comments**
CPU utilisation percentage.	Float	**cpu** - <*CPU number*> or *all* (default) **type** - possible values: *idle*, *nice*, *user* (default), *system* (default for Windows), *iowait*, *interrupt*, *softirq*, *steal*, *guest* (on Linux kernels 2.6.24 and above), *guest_nice* (on Linux kernels 2.6.33 and above). Parameters *user* and *nice* time no longer include *guest* time and *guest_nice* time since Zabbix 3.0.14. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*	Example: ⇒ system.cpu.util[0,user,avg5] Old naming: *system.cpu.idleX, system.cpu.niceX, system.cpu.systemX, system.cpu.userX*
system.hostname[<type>]			
System host name.	String	**type** (Windows only, must not be used on other systems) - possible values: *netbios* (default) or *host*	The value is acquired by either GetComputerName() (for **netbios**) or gethostname() (for **host**) functions on Windows and by "hostname" command on other systems. Examples of returned values: *on Linux*: ⇒ system.hostname → linux-w7x1 ⇒ system.hostname → www.zabbix.com *on Windows*: ⇒ system.hostname → WIN-SERV2008-I6 ⇒ system.hostname[host] → Win-Serv2008-I6LonG The `type` parameter for this item is supported since Zabbix **1.8.6**. See also a more detailed description.
system.hw.chassis[<info>]			
Chassis information.	String	**info** - one of full (default), model, serial, type or vendor	Example: system.hw.chassis[full] Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXX Desktop] This key depends on the availability of the SMBIOS table. Will try to read the DMI table from sysfs, if sysfs access fails then try reading directly from memory. **Root permissions** are required because the value is acquired by reading from sysfs or memory. Supported since Zabbix agent version 2.0.
system.hw.cpu[<cpu>,<info>]			
CPU information.	String or integer	**cpu** - <*CPU number*> or *all* (default) **info** - possible values: *full* (default), *curfreq*, *maxfreq*, *model* or *vendor*	Example: ⇒ system.hw.cpu[0,vendor] → AuthenticAMD Gathers info from /proc/cpuinfo and /sys/devices/system/cpu/[cpunum]/cpufreq/cpuinfo_max_freq. If a CPU number and *curfreq* or *maxfreq* is specified, a numeric value is returned (Hz). Supported since Zabbix agent version 2.0.
system.hw.devices[<type>]			
Listing of PCI or USB devices.	Text	**type** - *pci* (default) or *usb*	Example: ⇒ system.hw.devices[pci] → 00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge [..] Returns the output of either lspci or lsusb utility (executed without any parameters) Supported since Zabbix agent version 2.0.
system.hw.macaddr[<interface>,<format>]			

Key			
Description	**Return value**	**Parameters**	**Comments**
Listing of MAC addresses.	String	**interface** - *all* (default) or a regular expression **format** - *full* (default) or *short*	Lists MAC adresses of the interfaces whose name matches the given `interface` regexp (*all* lists for all interfaces). Example: ⇒ system.hw.macaddr["eth0$",full] → [eth0] 00:11:22:33:44:55 If `format` is specified as *short*, interface names and identical MAC addresses are not listed. Supported since Zabbix agent version 2.0.
system.localtime[<type>]			
System time.	Integer - with `type` as *utc* String - with `type` as *local*	**type** - possible values: *utc* - (default) the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds. *local* - the time in the 'yyyy-mm-dd,hh:mm:ss.nnn,+hh:mm' format	Parameters for this item are supported since Zabbix agent version 2.0. Example: ⇒ system.localtime[local] → create an item using this key and then use it to display host time in the *Clock* screen element.
system.run[command,<mode>]			
Run specified command on the host.	Text result of the command 1 - with `mode` as *nowait* (regardless of command result)	**command** - command for execution **mode** - possible values: *wait* - wait end of execution (default), *nowait* - do not wait	Up to 512KB of data can be returned, including trailing whitespace that is truncated. To be processed correctly, the output of the command must be text. Example: ⇒ system.run[ls -l /] → detailed file list of root directory. *Note*: To enable this functionality, agent configuration file must contain EnableRemoteCommands=1 option. *Note*: The return value of the item is standard output together with standard error produced by command. *Note:* Empty result is allowed starting with Zabbix 2.4.0. See also: Command execution.
system.stat[resource,<type>]			
System statistics.	Integer or float	**ent** - number of processor units this partition is entitled to receive (float) **kthr,<type>** - information about kernel thread states: *r* - average number of runnable kernel threads (float) *b* - average number of kernel threads placed in the Virtual Memory Manager wait queue (float) **memory,<type>** - information about the usage of virtual and real memory: *avm* - active virtual pages (integer) *fre* - size of the free list (integer) **page,<type>** - information about page faults and paging activity: *fi* - file page-ins per second (float) *fo* - file page-outs per second (float) *pi* - pages paged in from paging space (float) *po* - pages paged out to paging space (float) *fr* - pages freed (page replacement) (float) *sr* - pages scanned by page-replacement algorithm (float) **faults,<type>** - trap and interrupt rate: *in* - device interrupts (float) *sy* - system calls (float) *cs* - kernel thread context switches (float) **cpu,<type>** - breakdown of percentage usage of processor time: *us* - user time (float) *sy* - system time (float) *id* - idle time (float) *wa* - idle time during which the system had outstanding disk/NFS I/O request(s) (float) *pc* - number of physical processors consumed (float) *ec* - the percentage of entitled capacity consumed (float) *lbusy* - indicates the percentage of logical processor(s) utilization that occurred while executing at the user and system level (float) *app* - indicates the available physical processors in the shared pool (float) **disk,<type>** - disk statistics: *bps* - indicates the amount of data transferred (read or written) to the drive in bytes per second (integer) *tps* - indicates the number of transfers per second that were issued to the physical disk/tape (float) This item is supported since Zabbix **1.8.1.**	
system.sw.arch			

Key			
Description	**Return value**	**Parameters**	**Comments**
Software architecture information.	String		Example: ⇒ system.sw.arch → i686 Info is acquired from uname() function. Supported since Zabbix agent version 2.0.
system.sw.os[<info>]			
Operating system information.	String	**info** - possible values: *full* (default), *short* or *name*	Example: ⇒ system.sw.os[short]→ Ubuntu 2.6.35-28.50-generic 2.6.35.11 Info is acquired from (note that not all files are present in all distributions): /proc/version (*full*) /proc/version_signature (*short*) /etc/issue.net (*name*) Supported since Zabbix agent version 2.0.
system.sw.packages[<package>,<manager>,<format>]			
Listing of installed packages.	Text	**package** - *all* (default) or a regular expression **manager** - *all* (default) or a package manager **format** - *full* (default) or *short*	Lists (alphabetically) installed packages whose name matches the given `package` regexp (*all* lists them all). Example: ⇒ system.sw.packages[mini,dpkg,short] → python-minimal, python2.6-minimal, ubuntu-minimal Supported package managers (executed command): dpkg (dpkg --get-selections) pkgtool (ls /var/log/packages) rpm (rpm -qa) pacman (pacman -Q) If `format` is specified as *full*, packages are grouped by package managers (each manager on a seperate line beginning with its name in square brackets). If `format` is specified as *short*, packages are not grouped and are listed on a single line. Supported since Zabbix agent version 2.0.
system.swap.in[<device>,<type>]			
Swap in (from device into memory) statistics.	Integer	**device** - device used for swapping (default is *all*) **type** - possible values: *count* (number of swapins), *sectors* (sectors swapped in), *pages* (pages swapped in). See supported by platform for details on defaults.	Example: ⇒ system.swap.in[,pages] The source of this information is: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
system.swap.out[<device>,<type>]			
Swap out (from memory onto device) statistics.	Integer	**device** - device used for swapping (default is *all*) **type** - possible values: *count* (number of swapouts), *sectors* (sectors swapped out), *pages* (pages swapped out). See supported by platform for details on defaults.	Example: ⇒ system.swap.out[,pages] The source of this information is: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
system.swap.size[<device>,<type>]			
Swap space size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**device** - device used for swapping (default is *all*) **type** - possible values: *free* (free swap space, default), *pfree* (free swap space, in percent), *pused* (used swap space, in percent), *total* (total swap space), *used* (used swap space)	Example: ⇒ system.swap.size[,pfree] → free swap space percentage If *device* is not specified Zabbix agent will only take into account swap devices (files), physical memory will be ignored. For example, on Solaris systems *swap -s* command includes a portion of physical memory and swap devices (unlike *swap -l*). Note that this key might report incorrect percentage on virtualized (VMware ESXi, VirtualBox) Windows platforms. In this case use perf_counter[\700(_Total)\702] key to obtain correct swap usage data. Old naming: *system.swap.free*, *system.swap.total*
system.uname			

Key			
Description	**Return value**	**Parameters**	**Comments**
Identification of the system.	String		Example of returned value (Unix): FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386 Example of returned value (Windows): Windows ZABBIX-WIN 6.0.6001 Microsoft® Windows Server® 2008 Standard Service Pack 1 x86 On Unix since Zabbix 2.2.0 the value for this item is obtained with uname() system call. Previously it was obtained by invoking “uname -a”. The value of this item might differ from the output of “uname -a” and does not include additional information that “uname -a” prints based on other sources. On Windows since Zabbix 3.0 the value for this item is obtained from Win32_OperatingSystem and Win32_Processor WMI classes. Previously it was obtained from volatile Windows APIs and undocumented registry keys. The OS name (including edition) might be translated to the user's display language. On some versions of Windows it contains trademark symbols and extra spaces. Note that on Windows the item returns OS architecture, whereas on Unix it returns CPU architecture.
system.uptime			
System uptime in seconds.	Integer		In item configuration, use **s** or **uptime** units to get readable values.
system.users.num			
Number of users logged in.	Integer		**who** command is used on the agent side to obtain the value.
vfs.dev.read[<device>,<type>,<mode>]			
Disk read statistics.	Integer - with `type` in *sectors*, *operations*, *bytes* Float - with `type` in *sps*, *ops*, *bps*	**device** - disk device (default is *all*) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* This parameter must be specified, since defaults differ under various OSes. *sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*. This parameter is supported only with `type` in: sps, ops, bps.	Default values of 'type' parameter for different OSes: AIX - operations FreeBSD - bps Linux - sps OpenBSD - operations Solaris - bytes Example: ⇒ vfs.dev.read[,operations] *sps*, *ops* and *bps* on supported platforms used to be limited to 8 devices (7 individual and one *all*). Since Zabbix 2.0.1 this limit is 1024 devices (1023 individual and one for *all*). If default *all* is used for the first parameter then the key will return summary statistics, including all block devices like sda, sbd and their partitions (sda1, sda2, sdb3...) and multiple devices (MD raid) based on those block devices/partitions and logical volumes (LVM) based on those block devices/partitions. In such cases returned values should be considered only as relative value (dynamic in time) but not as absolute values. Supports LVM since Zabbix 1.8.6. Only relative device names could be used (for example, **sda**) until Zabbix 1.8.6. Since then, an optional **/dev/** prefix may be used (for example, **/dev/sda**). Old naming: *io[*]*
vfs.dev.write[<device>,<type>,<mode>]			

Key			
Description	**Return value**	**Parameters**	**Comments**
Disk write statistics.	Integer - with `type` in *sectors*, *operations*, *bytes* Float - with `type` in *sps*, *ops*, *bps*	**device** - disk device (default is *all*) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* This parameter must be specified, since defaults differ under various OSes. *sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*. This parameter is supported only with `type` in: sps, ops, bps.	Default values of 'type' parameter for different OSes: AIX - operations FreeBSD - bps Linux - sps OpenBSD - operations Solaris - bytes Example: ⇒ vfs.dev.write[,operations] *sps*, *ops* and *bps* on supported platforms used to be limited to 8 devices (7 individual and one *all*). Since Zabbix 2.0.1 this limit is 1024 (1023 individual and one for *all*). If default *all* is used for the first parameter then the key will return summary statistics, including all block devices like sda, sbd and their partitions (sda1, sda2, sdb3...) and multiple devices (MD raid) based on those block devices/partitions and logical volumes (LVM) based on those block devices/partitions. In such cases returned values should be considered only as relative value (dynamic in time) but not as absolute values. Supports LVM since Zabbix 1.8.6. Only relative device names could be used (for example, **sda**) until Zabbix 1.8.6. Since then, an optional **/dev/** prefix may be used (for example, **/dev/sda**). Old naming: *io[*]*
vfs.file.cksum[file]			
File checksum, calculated by the UNIX cksum algorithm.	Integer	**file** - full path to file	Example: ⇒ vfs.file.cksum[/etc/passwd] Example of returned value: 1938292000 Old naming: *cksum* The file size limit depends on large file support.
vfs.file.contents[file,<encoding>]			
Retrieving contents of a file.	Text	**file** - full path to file **encoding** - code page identifier	Returns an empty string if the file is empty or contains LF/CR characters only. Example: ⇒ vfs.file.contents[/etc/passwd] This item is limited to files no larger than 64 Kbytes. Supported since Zabbix agent version 2.0.
vfs.file.exists[file]			
Checks if file exists.	0 - not found 1 - regular file or a link (symbolic or hard) to regular file exists	**file** - full path to file	Example: ⇒ vfs.file.exists[/tmp/application.pid] The return value depends on what S_ISREG POSIX macro returns. The file size limit depends on large file support.
vfs.file.md5sum[file]			
MD5 checksum of file.	Character string (MD5 hash of the file)	**file** - full path to file	Example: ⇒ vfs.file.md5sum[/usr/local/etc/zabbix_agentd.conf] Example of returned value: b5052decb577e0fffd622d6ddc017e82 The file size limit (64 MB) for this item was removed in version 1.8.6. The file size limit depends on large file support.

Key			
Description	**Return value**	**Parameters**	**Comments**
vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]			
Find string in a file.	The line containing the matched string, or as specified by the optional `output` parameter	**file** - full path to file **regexp** - GNU regular expression **encoding** - code page identifier **start line** - the number of first line to search (first line of file by default). **end line** - the number of last line to search (last line of file by default). **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched text while an **\N** (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).	Only the first matching line is returned. An empty string is returned if no line matched the expression. Content extraction using the `output` parameter takes place on the agent. The `start line`, `end line` and `output` parameters are supported from version 2.2. Examples: ⇒ vfs.file.regexp[/etc/passwd,zabbix] ⇒ vfs.file.regexp[/path/to/some/file,"([0-9]+)$",,3,5,\1] ⇒ vfs.file.regexp[/etc/passwd,^zabbix:.:([0-9]+),,,,\1] → getting the ID of user *zabbix*
vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]			
Find string in a file.	0 - match not found 1 - found	**file** - full path to file **regexp** - GNU regular expression **encoding** - code page identifier **start line** - the number of first line to search (first line of file by default). **end line** - the number of last line to search (last line of file by default).	The `start line` and `end line` parameters are supported from version 2.2. Example: ⇒ vfs.file.regmatch[/var/log/app.log,error]
vfs.file.size[file]			
File size (in bytes).	Integer	**file** - full path to file	The file must have read permissions for user *zabbix*. Example: ⇒ vfs.file.size[/var/log/syslog] The file size limit depends on large file support.
vfs.file.time[file,<mode>]			
File time information.	Integer (Unix timestamp)	**file** - full path to the file **mode** - possible values: *modify* (default) - modification time, *access* - last access time, *change* - last change time	Example: ⇒ vfs.file.time[/etc/passwd,modify] The file size limit depends on large file support.
vfs.fs.discovery			
List of mounted filesystems. Used for low-level discovery.	JSON object		Supported since Zabbix agent version 2.0. {#FSDRIVETYPE} macro is supported on Windows since Zabbix agent version 3.0.
vfs.fs.inode[fs,<mode>]			
Number or percentage of inodes.	Integer - for number Float - for percentage	**fs** - filesystem **mode** - possible values: *total* (default), *free*, *used*, *pfree* (free, percentage), *pused* (used, percentage)	Example: ⇒ vfs.fs.inode[/,pfree] Old naming: *vfs.fs.inode.free[*]*, *vfs.fs.inode.pfree[*]*, *vfs.fs.inode.total[*]*
vfs.fs.size[fs,<mode>]			
Disk space in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**fs** - filesystem **mode** - possible values: *total* (default), *free*, *used*, *pfree* (free, percentage), *pused* (used, percentage)	In case of a mounted volume, disk space for local file system is returned. Example: ⇒ vfs.fs.size[/tmp,free] Reserved space of a file system is taken into account and not included when using the *free* mode. Old naming: *vfs.fs.free[*]*, *vfs.fs.total[*]*, *vfs.fs.used[*]*, *vfs.fs.pfree[*]*, *vfs.fs.pused[*]*
vm.memory.size[<mode>]			

Key			
Description	**Return value**	**Parameters**	**Comments**
Memory size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**mode** - possible values: *total* (default), *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *wired*, *used*, *pused* (used, percentage), *available*, *pavailable* (available, percentage)	This item accepts three categories of parameters: 1) *total* - total amount of memory; 2) platform-specific memory types: *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *wired*; 3) user-level estimates on how much memory is used and available: *used*, *pused*, *available*, *pavailable*. See a more detailed description of vm.memory.size parameters. Old naming: *vm.memory.buffers*, *vm.memory.cached*, *vm.memory.free*, *vm.memory.shared*, *vm.memory.total*
web.page.get[host,<path>,<port>]			
Get content of web page.	Web page source as text (including headers)	**host** - hostname **path** - path to HTML document (default is /) **port** - port number (default is 80)	Returns an empty string on fail. Example: ⇒ web.page.get[www.zabbix.com,index.php,80]
web.page.perf[host,<path>,<port>]			
Loading time of full web page (in seconds).	Float	**host** - hostname **path** - path to HTML document (default is /) **port** - port number (default is 80)	Returns 0 on fail. Example: ⇒ web.page.perf[www.zabbix.com,index.php,80]
web.page.regexp[host,<path>,<port>,<regexp>,<length>,<output>]			
Find string on a web page.	The matched string, or as specified by the optional `output` parameter	**host** - hostname **path** - path to HTML document (default is /) **port** - port number (default is 80) **regexp** - GNU regular expression **length** - maximum number of characters to return **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched text while an **\N** (where N=1…9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).	Returns an empty string if no match was found or on fail. Content extraction using the `output` parameter takes place on the agent. The `output` parameter is supported from version 2.2. Example: ⇒ web.page.regexp[www.zabbix.com,index.php,80,OK,2]

A Linux-specific note. Zabbix agent must have read-only access to filesystem */proc*. Kernel patches from www.grsecurity.org limit access rights of non-privileged users.

Available encodings

The `encoding` parameter is used to specify encoding for processing corresponding item checks, so that data acquired will not be corrupted. For a list of supported encodings (code page identifiers), please consult respective documentation, such as documentation for libiconv (GNU Project) or Microsoft Windows SDK documentation for “Code Page Identifiers”.

If empty `encoding` is passed, then UTF-8 (default locale for newer Unix/Linux distributions, see your system's settings) or ANSI with system-specific extension (Windows) is used by default.

Troubleshooting agent items

1. If used with passive agent, *Timeout* value in server configuration may need to be higher than *Timeout* in the agent configuration file. Otherwise the item may not get any value because the server request to agent timed out first.