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 supported by Zabbix agent 2
- Item keys specific for Windows agent
- Minimum permission level for Windows agent items

**Mandatory and optional parameters**

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent.hostname</td>
<td>Agent host name.</td>
<td>String</td>
<td>Returns:</td>
<td>As passive check - the name of the first host listed in the Hostname parameter of the agent configuration file; As active check - the name of the current hostname.</td>
</tr>
<tr>
<td>agent.ping</td>
<td>Agent availability check.</td>
<td>Nothing - unavailable 1 - available</td>
<td></td>
<td>Use the nodata() trigger function to check for host unavailability.</td>
</tr>
<tr>
<td>agent.version</td>
<td>Version of Zabbix agent.</td>
<td>String</td>
<td></td>
<td>Example of returned value: 1.8.2</td>
</tr>
<tr>
<td>kernel.maxfiles</td>
<td>Maximum number of opened files supported by OS.</td>
<td>Integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>kernel.maxproc</td>
<td>Maximum number of processes supported by OS.</td>
<td>Integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>log[file,&lt;regexp&gt;,&lt;encoding&gt;,&lt;maxlines&gt;,&lt;mode&gt;,&lt;output&gt;,&lt;maxdelay&gt;,&lt;options&gt;]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Description
logrt[file_regexp,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>]

#### Return value
- **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 analyze. Value: 0 - (default) never ignore log file lines; > 0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it!
- **mode**: possible values: all (default), skip - ignore log file lines
- **maxdelay**: maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it!
- **output**: an optional output formatting template. The 0 escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an (N) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).
- **options**: additional options: mtime-noreread - non-unique records, reread only if the file size changes (ignore modification time change). This parameter is deprecated since 5.0.2, because now mtime is ignored.

#### Parameters
- **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 analyze. Value: 0 - (default) never ignore log file lines; > 0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it!
- **mode**: possible values: all (default), skip - ignore log file lines
- **maxdelay**: maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it!
- **output**: an optional output formatting template. The 0 escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an (N) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).
- **options**: additional options: mtime-noreread - non-unique records, reread only if the file size changes (ignore modification time change). This parameter is deprecated since 5.0.2, because now mtime is ignored.

#### Comments
- The item must be configured as an active check.
- If file is missing or permissions do not allow access, item turns unsupported.
- See also additional information on log monitoring.

### Description
log.count[file,regexp,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>]

#### Return value
- **file**: full path and name of log file
- **regexp**: regular expression describing the required pattern
- **encoding**: code page identifier
- **maxproclines**: maximum number of new lines per second the agent will analyze. Value: 0 - (default) never ignore log file lines; > 0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it!
- **mode**: possible values: all (default), skip - ignore log file lines
- **maxdelay**: maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it!
- **options**: additional options: mtime-noreread - non-unique records, reread only if the file size changes (ignore modification time change). This parameter is deprecated since 5.0.2, because now mtime is ignored.

#### Parameters
- **file**: full path and name of log file
- **regexp**: regular expression describing the required pattern
- **encoding**: code page identifier
- **maxproclines**: maximum number of new lines per second the agent will analyze. Value: 0 - (default) never ignore log file lines; > 0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it!
- **mode**: possible values: all (default), skip - ignore log file lines
- **maxdelay**: maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it!
- **options**: additional options: mtime-noreread - non-unique records, reread only if the file size changes (ignore modification time change). This parameter is deprecated since 5.0.2, because now mtime is ignored.

#### Comments
- The item must be configured as an active check.
- If file is missing or permissions do not allow access, item turns unsupported.
- See also additional information on log monitoring.

### Description
Log file monitoring.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
</table>
| Log   | Log file monitoring. | file - full path and name of log file<br>regexp - regular expression describing the required pattern<br>encoding - code page identifier<br>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<br>mode (since version 2.0): possible values: all (default), skip - skip processing of older data (affects only newly created items).<br>output (since version 2.2): an optional output formatting template. The 0 escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an (N) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).<br>maxdelay (since version 3.2): maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it!
| options (since version 4.4.7): additional options: mtime-noreread - non-unique records, reread only if the file size changes (ignore modification time change). This parameter is deprecated since 5.0.2, because now mtime is ignored. | The item must be configured as an active check. If file is missing or permissions do not allow access, item turns unsupported. | See also additional information on log monitoring. |

This item is not supported for Windows Event Log. Supported since Zabbix 3.2.0.

Content extraction using the output parameter takes place on the agent.

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.

Using output parameter for extracting a number from log record:

⇒ log[/var/log/syslog]<br>⇒ log[/var/log/syslog,error]<br>⇒ log[/home/zabbix/logs/logfile,,,100]

Examples:

⇒ log[/var/log/syslog]<br>⇒ log[/var/log/syslog,error]<br>⇒ log[/home/zabbix/logs/logfile,,,100]

Content extraction using the output parameter takes place on the agent.

Examples:

⇒ log[/app1/app.log,"task run \[0-9\.]+ sec, processed \[0-9\]+ records, \[0-9\]+ errors\",\"\"] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send only '6080' to server. Because a numeric value is being sent, the "Type of information" for this item can be set to "Numeric (unsigned)" and the value can be used in graphs, triggers etc.

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\",\"\"] → 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.

See also additional information on log monitoring.

Supported since Zabbix 3.2.0.
### Logrt.count

<table>
<thead>
<tr>
<th>Description</th>
<th>Parameters</th>
<th>Return value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log rotation monitoring with log file support.</td>
<td>file_regexp - absolute path to file and file name described by a regular expression. Note that only the file name is a regular expression</td>
<td>Integer</td>
<td>The item must be configured as an active check. Log rotation is based on the last modification time of files.</td>
</tr>
</tbody>
</table>

The item must be configured as an active check. Log rotation is based on the last modification time of files.

Note that logrt is designed to work with only one currently active log file, with several other matching inactive files rotated. If, for example, a directory has many active log files, a separate logrt item should be created for each one. Otherwise if one logrt item picks up too many files it may lead to exhausted memory and a crash of monitoring.

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\[/app1/^test.*log$,"(\[0-9-:]+) task run (\[0-9.]+) sec, processed (\[0-9\]+) records, (\[0-9\]+) errors",",,,"]` → will match a log record “2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors” and send only ‘6080′ to server. Because a numeric value is being sent, the “Type of information” for this item can be set to “Numeric (unsigned)” and the value can be used in graphs, triggers etc.

- `logrt\[/app1/"test\*.log","(\[0-9-:]+) task run (\[0-9.]+) sec, processed (\[0-9\]+) records, (\[0-9\]+) errors",",,,"]` → will match a log record “2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors” and send only ‘6080′ to server. Because a numeric value is being sent, the “Type of information” for this item can be set to “Numeric (unsigned)” and the value can be used in graphs, triggers etc.

Using output parameter for extracting a number from log record:

- `logrt\[/api/"test:\*","(\[0-9-:]+) task run (\[0-9.]+) sec, processed (\[0-9\]+) records, (\[0-9\]+) errors",",,,"]` → will match a log record “2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors” and send only ‘6080′ to server. Because a numeric value is being sent, the “Type of information” for this item can be set to “Numeric (unsigned)” and the value can be used in graphs, triggers etc.

Using output parameter for rewriting log record before sending to server:

- `logrt\[/api/"test:\*","(\[0-9-:]+) task run (\[0-9.]+) sec, processed (\[0-9\]+) records, (\[0-9\]+) errors",",,,"]` → 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 task run 6.08 sec, processed 6080 records, 0 errors, DURATION: 6.08” to server.

See also additional information on log monitoring.

#### modbus.get

<table>
<thead>
<tr>
<th>Description</th>
<th>Parameters</th>
<th>Return value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reads Modbus data.</td>
<td>endpoint - endpoint defined as protocol://connection_string slave ID - slave ID function - Modbus function address - address of first registry, coil or input count - number of records to read type - type of data endianness - endianness configuration offset - number of registers, starting from 'address', the results of which will be discarded. See a detailed description of parameters.</td>
<td>JSON object</td>
<td>Supported since Zabbix 5.2.0.</td>
</tr>
</tbody>
</table>

Supported since Zabbix 5.2.0.

#### net.dns

<table>
<thead>
<tr>
<th>Description</th>
<th>Parameters</th>
<th>Return value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reads NetDNS data.</td>
<td>&lt;ip&gt;,name,&lt;&lt;timeout&gt;&gt;,&lt;&lt;count&gt;&gt;,&lt;&lt;protocol&gt;&gt;</td>
<td></td>
<td>Supported since Zabbix 5.2.0.</td>
</tr>
</tbody>
</table>
### net.dns.record

**Key**
- `ip`, `name`, `type`, `timeout`, `count`, `protocol`

**Description**
Performs a DNS query.

**Parameters**
- `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` - timeout for the request in seconds (default is 1 second)
- `count` - number of tries for the request (default is 2)
- `protocol` (since version 3.0) - the protocol used to perform DNS queries: udp (default) or tcp

**Example:**
```plaintext
⇒ net.dns[8.8.8.8,example.com,MX,2,1]
```

**Comments**
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.

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.if.collision

**Key**
- `if`

**Description**
Number of out-of-window collisions.

**Parameters**
- `if` - network interface name

**Example:**
```plaintext
⇒ net.if.collision[eth0]
```

### net.if.discovery

**Key**
- `if`, `<mode>`

**Description**
List of network interfaces. Used for low-level discovery.

**Parameters**
- `if` - network interface name

**Example:**
```plaintext
⇒ net.if.discovery
```

### net.if.in

**Key**
- `if`, `<mode>`

**Description**
Incoming traffic statistics on network interface.

**Parameters**
- `if` - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows)
- `mode` - possible values: `bytes` - number of bytes (default)
- `packets` - number of packets
- `errors` - number of errors
- `dropped` - number of dropped packets
- `overruns` (fifo) - the number of FIFO buffer errors
- `frame` - the number of packet framing errors
- `compressed` - the number of compressed packets transmitted or received by the device driver
- `multicast` - the number of multicast frames received by the device driver

**Example:**
```plaintext
⇒ net.if.in[eth0]
```

### net.if.out

**Key**
- `if`, `<mode>`

**Description**
Outgoing traffic statistics on network interface.

**Parameters**
- `if` - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows)
- `mode` - possible values:
- `bytes` - number of bytes (default)
- `packets` - number of packets
- `errors` - number of errors
- `dropped` - number of dropped packets
- `overruns` (fifo) - the number of FIFO buffer errors
- `collisions` (colls) - the number of collisions detected on the interface carrier - the number of carrier losses detected by the device driver
- `compressed` - the number of compressed packets transmitted by the device driver

**Example:**
```plaintext
⇒ net.if.out[eth0]
```

### net.if.total

**Key**
- `if`, `<mode>`

**Description**
Change per second

**Parameters**
- `if` - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows)
- `mode` - possible values:

**Example:**
```plaintext
⇒ net.if.total[eth0]
```
### net.tcp.listen[<port>]
Checks if the TCP port is in LISTEN state.

- **Key**: port - TCP port number

- **Example**: net.tcp.listen[80]

- **Note**: On Windows supported since Zabbix agent version 1.8.4

### net.tcp.service[service, <ip>, <port>]
Checks if the service is running and accepting TCP connections.

- **Key**: service - either of: ssh, ldap, smtp, ftp, http, pop, nntp, imap, tcp, https, telnet (see details)
- **Example**: net.tcp.service[ftp,,45]

### net.tcp.service.perf[service, <ip>, <port>]
Checks performance of TCP service.

- **Key**: service - either of: ssh, ldap, smtp, ftp, http, pop, nntp, imap, tcp, https, telnet (see details)

### net.udp.listen[<port>]
Checks if this UDP port is in LISTEN state.

- **Key**: port - UDP port number

- **Example**: net.udp.listen[68]

### net.udp.service[service, <ip>, <port>]
Checks if service is running and responding to UDP requests.

- **Key**: service - ntp (see details)

### Known issues with net.udp.service
- 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 for checks like these.
- Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2.
### Checks performance of UDP service.

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 - service is down seconds</td>
<td>service - ntp (see details)</td>
<td>Example: net.udp.service.perf[ntp]→ can be used to test response time from NTP service.</td>
</tr>
<tr>
<td></td>
<td>- the number of seconds spent</td>
<td>ip - IP address (default is 127.0.0.1)</td>
<td>This item is supported since Zabbix 3.0.0, but ntp service was available for net.tcp.service() item in prior versions.</td>
</tr>
<tr>
<td></td>
<td>waiting for response from the</td>
<td>port - port number (by default standard service port number is used)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### proc.cpu.util(<name>, <user>, <type>, <cmdline>, <mode>, <zone>)

**Process CPU utilization percentage.**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>process name (default is all processes)</td>
<td>Float</td>
<td>name - process name (default is all processes)</td>
<td>Examples: proc.cpu.util(root) → CPU utilization of all processes running under the “root” user</td>
</tr>
<tr>
<td>user</td>
<td>user name (default is all users)</td>
<td></td>
<td>user - user name (default is all users)</td>
<td>proc.cpu.util[zabbix_server,zabbix] → CPU utilization of all zabbix_server processes running under the zabbix user</td>
</tr>
<tr>
<td>type</td>
<td>CPU utilization type:</td>
<td></td>
<td>type - CPU utilization type:</td>
<td>The returned value is based on single CPU core utilization percentage. For example</td>
</tr>
<tr>
<td></td>
<td>(global) (default), user, system</td>
<td></td>
<td>(global) (default), user, system</td>
<td>CPU utilization of a process fully using two cores is 200%.</td>
</tr>
<tr>
<td>cmdline</td>
<td>filter by command line (it is a regular expression)</td>
<td>Float - with mode as avg</td>
<td>cmdline - filter by command line (it is a regular expression)</td>
<td>The process CPU utilization data is gathered by a collector which supports the</td>
</tr>
<tr>
<td>mode</td>
<td>data gathering mode: avg2 (default), avg5, avg25</td>
<td></td>
<td>mode - data gathering mode: avg2 (default), avg5, avg25</td>
<td>maximum of 1024 unique (by name, user and command line) queries. Queries not</td>
</tr>
<tr>
<td>zone</td>
<td>target zone: current (default), all.</td>
<td></td>
<td>zone - target zone: current (default), all. This parameter is supported on Solaris only.</td>
<td>accessed during the last 24 hours are removed from the collector.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: When setting the zone parameter to current (or default) in case the agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>has been compiled on a Solaris without zone support, but running on a newer Solaris</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>where zones are supported, then the agent will return NOTSUPPORTED (the agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>cannot limit results to only the current zone). However, all is supported in this case.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This key is supported since Zabbix 3.0.0 and is available on several platforms (see</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>items supported by platform).</td>
</tr>
</tbody>
</table>

#### proc.mem(<name>, <user>, <mode>, <cmdline>, <memtype>)

**Memory used by process in bytes.**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>process name (default is all processes)</td>
<td>Integer - with mode as max, min, sum</td>
<td>name - process name (default is all processes)</td>
<td>Examples: proc.mem[root] → memory used by all processes running under the “root” user</td>
</tr>
<tr>
<td>user</td>
<td>user name (default is all users)</td>
<td>Float - with mode as avg</td>
<td>user - user name (default is all users)</td>
<td>proc.mem[zabbix_server,zabbix] → memory used by all zabbix_server processes running under the zabbix user</td>
</tr>
<tr>
<td>mode</td>
<td>possible values: avg, max, min, sum (default)</td>
<td></td>
<td>mode - possible values: avg, max, min, sum (default)</td>
<td>proc.mem[oracle,max,oracleZABBIX] → memory used by the most memory-hungry process running under oracle having oracleZABBIX in its command line</td>
</tr>
<tr>
<td>cmdline</td>
<td>filter by command line (it is a regular expression)</td>
<td></td>
<td>cmdline - filter by command line (it is a regular expression)</td>
<td>Note: When several processes use shared memory, the sum of memory used by</td>
</tr>
<tr>
<td>memtype</td>
<td>type of memory used by process</td>
<td></td>
<td>memtype - type of memory used by process</td>
<td>processes may result in large, unrealistic values.</td>
</tr>
</tbody>
</table>

#### proc.num(<name>, <user>, <state>, <cmdline>, <zone>)

**The number of processes.**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>process name (default is all processes)</td>
<td>Integer</td>
<td>name - process name (default is all processes)</td>
<td>Examples: proc.num[mysql] → number of processes running under the mysql user</td>
</tr>
<tr>
<td>user</td>
<td>user name (default is all users)</td>
<td></td>
<td>user - user name (default is all users)</td>
<td>proc.num[apache2,www-data] → number of apache2 processes running under the www-data user</td>
</tr>
<tr>
<td>state</td>
<td>(disk and trace options since version 3.4.0) -</td>
<td></td>
<td>state (disk and trace options since version 3.4.0) - possible values:</td>
<td>proc.num[oracle,sleep,oracleZABBIX] → number of processes in sleep state running under oracle having oracleZABBIX in its command line</td>
</tr>
<tr>
<td></td>
<td>possible values: all (default), disk -</td>
<td></td>
<td>(disk and trace options since version 3.4.0) - possible values: all (default), disk - interruptible sleep, run - running, sleep - interruptible sleep, trace - stopped, zomb - zombie</td>
<td>See notes on selecting processes with name and cmdline parameters (Linux-specific).</td>
</tr>
<tr>
<td></td>
<td>interruptible sleep, run - running, sleep -</td>
<td></td>
<td>state (disk and trace options since version 3.4.0) - possible values:</td>
<td>On Windows, only the name and user parameters are supported.</td>
</tr>
<tr>
<td></td>
<td>interruptible sleep, trace - stopped, zomb -</td>
<td></td>
<td>(disk and trace options since version 3.4.0) - possible values: all (default), disk - interruptible sleep, run - running, sleep - interruptible sleep, trace - stopped, zomb - zombie</td>
<td>When this item is invoked from the command line and contains a command line</td>
</tr>
<tr>
<td></td>
<td>zombie</td>
<td></td>
<td>state (disk and trace options since version 3.4.0) - possible values:</td>
<td>parameter (e.g. using the agent test mode: zabbix_agentd -t proc.num[, , , apache2]), one extra process will be counted, as the agent will count itself.</td>
</tr>
<tr>
<td></td>
<td>zone - target zone: current (default), all.</td>
<td></td>
<td>zone - target zone: current (default), all. This parameter is supported on Solaris only.</td>
<td>Note that when setting the zone parameter to current (or default) in case the agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>has been compiled on a Solaris without zone support, but running on a newer Solaris</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>where zones are supported, then the agent will return NOTSUPPORTED (the agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>cannot limit results to only the current zone). However, all is supported in this case.</td>
</tr>
</tbody>
</table>

#### sensor(device,sensor,<mode>]

- Note that when setting the zone parameter to current (or default) in case the agent has been compiled on a Solaris without zone support, but running on a newer Solaris where zones are supported, then the agent will return NOTSUPPORTED (the agent cannot limit results to only the current zone). However, all is supported in this case.
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware sensor reading.</td>
<td>Float</td>
<td>device - device name sensor - sensor name mode - possible values: avg, max, min (if this parameter is omitted, device and sensor are treated verbatim).</td>
<td></td>
<td>Returns /proc/sys/dev/sensors on Linux 2.4. Prior to Zabbix 1.8.4, the sensor[temp1] format was used.</td>
</tr>
<tr>
<td>List of detected CPUs/CPUs cores. Used for low-level discovery.</td>
<td>JSON object</td>
<td></td>
<td></td>
<td>Supported on all platforms since 2.4.0.</td>
</tr>
<tr>
<td>Device interrupts.</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of CPUs.</td>
<td>Integer type - possible values: online (default), max</td>
<td>Example: =&gt; system.cpu.num</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU utilization percentage.</td>
<td>Float</td>
<td>cpu - &lt;$CPU number&gt; or all (default) type - possible values: user (default), idle, nice, 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). See also platform-specific: details for this parameter mode - possible values: avg1 (one-minute average, default), avg5, avg15 logical_or_physical (since version 5.0.3; on AIX only) - possible values: logical (default), physical. This parameter is supported on AIX only.</td>
<td>Example: =&gt; system.cpu.util[0,user,avg5]</td>
<td>Old naming: system.cpu.idleX, system.cpu.niceX, system.cpu.systemX, system.cpu.userX</td>
</tr>
<tr>
<td>Chassis information.</td>
<td>String</td>
<td>info - one of full (default), model, serial, type or vendor</td>
<td></td>
<td>Example: system.hw.chassis[full] Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXX 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.</td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
<td>Return value</td>
<td>Parameters</td>
<td>Comments</td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>----------------------</td>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>system.hw.cpu[&lt;CPU number&gt; or all (default)] info</td>
<td>CPU information.</td>
<td>String or integer</td>
<td><code>cpu</code> - &lt;CPU number&gt; or all (default) info - possible values: full (default), curfreq, maxfreq, model or vendor</td>
<td>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.</td>
</tr>
<tr>
<td>system.hw.devices[&lt;type&gt;]</td>
<td>Listing of PCI or USB devices.</td>
<td>Text</td>
<td><code>type</code> (since version 2.0) - pci (default) or usb</td>
<td>Returns the output of either lspci or lsusb utility (executed without any parameters).</td>
</tr>
<tr>
<td>system.hw.macaddr[&lt;interface&gt;,&lt;format&gt;]</td>
<td>Listing of MAC addresses.</td>
<td>String</td>
<td><code>interface</code> - all (default) or a regular expression <code>format</code> - full (default) or short</td>
<td>Lists MAC addresses of the interfaces whose name matches the given interface regular expression (all lists for all interfaces). Example: <code>⇒ system.hw.macaddr[&quot;eth0&quot;,full] → [eth0] 00:11:22:33:44:55</code> If format is specified as short, interface names and identical MAC addresses are not listed. Supported since Zabbix agent version 2.0.</td>
</tr>
<tr>
<td>system.localtime[&lt;type&gt;]</td>
<td>System time.</td>
<td>Integer - with type as local</td>
<td><code>type</code> (since version 2.0) - 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,’ format</td>
<td>Must be used as a passive check only. Example: <code>⇒ system.localtime[local] → create an item using this key and then use it to display host time in the Clock dashboard widget</code>.</td>
</tr>
<tr>
<td>system.run[command,&lt;mode&gt;]</td>
<td>Run specified command on the host.</td>
<td>Text result of the command</td>
<td><code>command</code> - command for execution <code>mode</code> - possible values: wait - wait end of execution (default), nowait - do not wait</td>
<td>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: <code>⇒ system.run[ls -l] → detailed file list of root directory</code>. Note: system.run items are disabled by default. Learn how to enable them. The return value of the item is standard output together with standard error produced by command. The exit code is not checked. Empty result is allowed starting with Zabbix 2.4.0. See also: Command execution.</td>
</tr>
<tr>
<td>system.stat[resource,&lt;type&gt;]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
<td>Parameters</td>
<td>Return value</td>
<td>Comments</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
<td>------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td><strong>system.swap.in</strong> [:device:], [:type:]</td>
<td>Swap in (from device into memory) statistics.</td>
<td>device - device used for swapping (default is all) type - possible values: count (number of swaps), sectors (sectors swapped in), pages (pages swapped in). See also platform-specific details for this parameter.</td>
<td>Integer</td>
<td>Supported since Zabbix agent version 2.0.</td>
</tr>
<tr>
<td><strong>system.swap.out</strong> [:device:], [:type:]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>system.swap.in</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>system.swap.out</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>system.sw.os</strong> [[:info:]]</td>
<td>Operating system information.</td>
<td>info - possible values: full (default), short or name</td>
<td>String</td>
<td>Supported since Zabbix agent version 2.0.</td>
</tr>
<tr>
<td><strong>system.sw.packages</strong> [[:package:]], [[:manager:]], [[:format:]]</td>
<td>Listing of installed packages.</td>
<td>package - all (default) or a regular expression manager - all (default) or a package manager format - full (default) or short</td>
<td>Text</td>
<td>Lists (alphabetically) installed packages whose name matches the given package regular expression (all lists them all). Example: =&gt; 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 separate 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.</td>
</tr>
<tr>
<td><strong>system.sw.arch</strong></td>
<td>System architecture information.</td>
<td></td>
<td>Integer or float</td>
<td></td>
</tr>
<tr>
<td><strong>system.sw.os</strong> [[:short:]]</td>
<td>Operating system information.</td>
<td>info - possible values: full (default), short or name</td>
<td>String</td>
<td>Info is acquired from /etc/issue.net (PRETTY_NAME parameter from /etc/os-release on systems supporting it, or /etc/issue on older systems) Short name: /etc/issue.net (name)</td>
</tr>
<tr>
<td><strong>system.sw.packages</strong> [[:package:]], [[:manager:]], [[:format:]]</td>
<td>Listing of installed packages.</td>
<td>package - all (default) or a regular expression manager - all (default) or a package manager format - full (default) or short</td>
<td>Text</td>
<td>Lists (alphabetically) installed packages whose name matches the given package regular expression (all lists them all). Example: =&gt; 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 separate 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.</td>
</tr>
<tr>
<td><strong>system.swap.in</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>system.swap.out</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>system.swap.in</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>system.swap.out</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>system.swap.in</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>system.swap.out</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**See also** platform-specific details for this parameter.
### system.swap.size[<device>,<type>]

**Description**
Swap space size in bytes or in percentage from total.

**Parameters**
- device: device used for swapping (default is all)
- type: possible values: free (free swap space), default, pfree (free swap space, in percent), total (total swap space), used (used swap space)
- in percent: value for set as a percentage
- Total space: used swap space, in percent
- available: total swap space

**Return value**
- Integer - for bytes
- Float - for percentage

**Example**
```
$ system.swap.size[pfree] → free swap space percentage
```

**Comments**
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 -i).

Note that this key might report incorrect swap space size/percentage on virtualized (VMware ESXi, VirtualBox) Windows platforms. In this case you may use the `perf_counter[\700(_Total)\702]` key to obtain correct swap space percentage.

### system.uname

**Description**
Identification of the system.

**Parameters**
- Platform: platform for which the system configuration will be used

**Return value**
- String

**Example of returned value (Unix):**
```
FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov 1 08:02:36 EST 2002
```

**Example of returned value (Windows):**
```
Windows ZABBIX-WIN 6.0.6001 Microsoft® Windows Server® 2008 Standard Service Pack 1 x86
```

**Comments**
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

**Description**
System uptime in seconds.

**Parameters**
- in seconds

**Return value**
- Integer

**Example**
```
$ system.uptime → uptime in seconds
```

### system.users.num

**Description**
Number of users logged in.

**Parameters**
- in

**Return value**
- Integer

**Example**
```
$ system.users.num → number of users logged in
```

### vfs.dev.discovery

**Description**
List of block devices and their type. Used for low-level discovery.

**Parameters**
- JSON object

**Return value**
- JSON object

**Example**
```
$ vfs.dev.discovery → list of block devices and their type
```

### vfs.dev.read[<device>,<type>,<mode>]

**Description**
Disk read statistics.

**Parameters**
- device: disk device (default is all)  
- type: possible values: sectors, operations, bytes, sps, bps  
- mode: possible values: avg1 (one-minute average, default), avg5, avg15

**Return value**
- Integer - with type in sectors, operations, bytes  
- Float - with type in sps, bps

**Example**
```
$ vfs.dev.read[device, type, mode] → disk read statistics
```

### vfs.dev.write[<device>,<type>,<mode>]

**Description**
Disk write statistics.

**Parameters**
- device: disk device (default is all)  
- type: possible values: sectors, operations, bytes, sps, bps

**Return value**
- Integer - with type in sectors, operations, bytes  
- Float - with type in sps, bps

**Example**
```
$ vfs.dev.write[device, type, mode] → disk write statistics
```

---

https://www.zabbix.com/documentation/current/

Printed on 2021/10/14 07:58
## Parameters

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk write statistics.</td>
<td>Integer - with type in sectors, operations-bytes</td>
<td>file - disk device (default is all?) type - possible values: sectors, operations, bytes, s, s, bps</td>
<td>You may use relative device names (for example, sda) as well as an optional /dev prefix (for example, /dev/sda).</td>
</tr>
<tr>
<td>File checksum, calculated by the UNIX cksum algorithm.</td>
<td>Integer</td>
<td>file - full path to file</td>
<td>Example: <code>vfs.file.cksum[etc/password]</code> Example of returned value: <code>2658159705</code> The file size limit depends on large file support.</td>
</tr>
</tbody>
</table>
| Directory entry count. | Integer | dir - absolute path to directory regex_incl - regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value) regex_excl - regular expression describing the name pattern of the entity (file, directory, symbolic link) to exclude; don't exclude any if empty (default value) types_incl - directory entry types to count, possible values: file - regular file, dir - subdirectory, sym - symbolic link, sock - socket, bdev - block device, cdev - character device, fifo - FIFO, dev - device (default), dev - synchronous with "bdev", cdev-, all - all types (default), i.e. "dir,dir,sym,bdev,cdev,fifo", Multiple types must be separated with comma and quoted.
| Supported since Zabbix 4.0.0. |
| Directory size (in bytes). | Integer | dir - absolute path to directory regex_incl - regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value) regex_excl - regular expression describing the name pattern of the entity (file, directory, symbolic link) to exclude; don't exclude any if empty (default value) mode - possible values: apparent (default) - gets apparent file sizes rather than disk usage (acts as d - sb - d1r), disk - gets disk usage (acts as d as d - s - b1 r). Unlike du command, vfs.dir.size item takes hidden files in account when calculating directory size (acts as d as d - s - 1d1r). max_depth - maximum depth of subdirectories to traverse, -1 (default) - unlimited, 0 - no descending into subdirectories. |
| Supported since Zabbix 3.4.0. |
| Example: `vfs.file.cksum[etc/password]` Example of returned value: `1938292000` The file size limit depends on large file support. |

### Examples:

- `vfs.dir.size[/tmp,log]` - calculates size of all files in /tmp which contain 'log'
- `vfs.dir.size[tmp,log,-*]` - calculates size of all files in /tmp but excludes files containing "old"

### Environment variables:

- %APP_HOME%
- %HOME%
- %TEMP%

### Remarks:

- Neither directories nor symbolic links are counted.
- Symbolic links are not followed for directory traversal.
- On Windows, directory symlinks are skipped and hard links are counted only once.
- When filtering by size, only regular files have meaningful sizes. Under Linux and BSD, directories also have non-zero sizes (a few Kb typically). Devices have zero sizes, e.g. the size of /dev/sda does not reflect the respective partition size. Therefore, when using <min_size> and <max_size>, it is advisable to specify <types_incl> as "file", to avoid surprises.
- When filtering by size, only regular files have meaningful sizes. Under Linux and BSD, directories also have non-zero sizes (a few Kb typically). Devices have zero sizes, e.g. the size of /dev/sda does not reflect the respective partition size. Therefore, when using <min_size> and <max_size>, it is advisable to specify <types_incl> as "file", to avoid surprises.
- When filtering by size, only regular files have meaningful sizes. Under Linux and BSD, directories also have non-zero sizes (a few Kb typically). Devices have zero sizes, e.g. the size of /dev/sda does not reflect the respective partition size. Therefore, when using <min_size> and <max_size>, it is advisable to specify <types_incl> as "file", to avoid surprises.
- Only directories with at least read permission for zabbix user are calculated.
- On Windows any symlink is skipped and hard links are taken into account only once.
- With large directories or slow drives this item may time out due to the Timeout setting in agent and server/proxy configuration files. Increase the timeout values as necessary.
- Only directories with at least read permission for zabbix user are calculated.
- On Windows any symlink is skipped and hard links are taken into account only once.
- With large directories or slow drives this item may time out due to the Timeout setting in agent and server/proxy configuration files. Increase the timeout values as necessary.
- Only directories with at least read permission for zabbix user are calculated.
- On Windows any symlink is skipped and hard links are taken into account only once.
- With large directories or slow drives this item may time out due to the Timeout setting in agent and server/proxy configuration files. Increase the timeout values as necessary.
** vfs.fs.get

** vfs.fs.discovery

** vfs.fs.get

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists

** vfs.fs.file.time

** vfs.fs.exists
### Returned Values

<table>
<thead>
<tr>
<th>Function</th>
<th>Return Value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>zabbix.stats(ip,&lt;port&gt;)</code></td>
<td>JSON object</td>
<td><code>ip</code>&lt;br&gt;<code>&lt;port&gt;</code></td>
<td>Supported since Zabbix agent version 4.4.5.</td>
</tr>
<tr>
<td><code>vfs.fs.inode(fs,&lt;mode&gt;)</code></td>
<td>Integer - for number&lt;br&gt;Float - for percentage</td>
<td><code>fs - filesystem mode - possible values: total (default), free, used, pfree (free, percentage), used (used, percentage)</code></td>
<td>Example: <code>vfs.fs.inode[/pfree]</code></td>
</tr>
<tr>
<td><code>vfs.fs.size(fs,&lt;mode&gt;)</code></td>
<td>Integer - for bytes&lt;br&gt;Float - for percentage</td>
<td><code>fs - filesystem mode - possible values: total (default), free, used, pfree (free, percentage), used (used, percentage)</code></td>
<td>Example: <code>vfs.fs.size[/pfree]</code> In case of a mounted volume, disk space for local file system is returned. Reserved space of a file system is taken into account and not included when using the <code>free</code> mode.</td>
</tr>
<tr>
<td><code>vm.memory.size(&lt;mode&gt;)</code></td>
<td>Integer - for bytes&lt;br&gt;Float - for percentage</td>
<td><code>mode - possible values: total (default), active, anon, buffers, cached, exec, file, free, inactive, pinned, shared, slab, wired, used (used, percentage), available, pavailable (available, percentage)</code></td>
<td>This item accepts three categories of parameters: <code>1) total - total amount of memory; 2) platform-specific memory types: active, anon, buffers, cached, exec, file, free, inactive, pinned, shared, slab, wired; 3) user-level estimates on how much memory is used and available: used, pinned, available, pavailable.</code></td>
</tr>
<tr>
<td><code>web.page.get(host,&lt;path&gt;,&lt;port&gt;)</code></td>
<td></td>
<td><code>host - hostname or URL (as scheme://host:port/path, where only host is mandatory). Allowed URL schemes: http, https solutions. Missing scheme will be treated as http. If URL is specified path and port must be empty. Specifying user name/password when connecting to servers that require authentication, for example: http://user:password@www.example.com is only possible with cURL support. Punycode is supported in hostnames. path - path to HTML document (default is /) port - port number (default is 80 for HTTP)</code></td>
<td>This item turns unsupported if the resource specified in <code>host</code> does not exist or is unavailable. This item turns unsupported if the resource specified in <code>host</code> does not exist or is unavailable. This item turns unsupported if the resource specified in <code>host</code> does not exist or is unavailable. <code>host</code> can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled. <code>host</code> can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled.</td>
</tr>
<tr>
<td><code>web.page.perf(host,&lt;path&gt;,&lt;port&gt;)</code></td>
<td>Float</td>
<td><code>host - hostname or URL (as scheme://host:port/path, where only host is mandatory). Allowed URL schemes: http, https solutions. Missing scheme will be treated as http. If URL is specified path and port must be empty. Specifying user name/password when connecting to servers that require authentication, for example: http://user:password@www.example.com is only possible with cURL support. Punycode is supported in hostnames. path - path to HTML document (default is /) port - port number (default is 80 for HTTP)</code></td>
<td>This item turns unsupported if the resource specified in <code>host</code> does not exist or is unavailable. <code>host</code> can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled. <code>host</code> can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled.</td>
</tr>
<tr>
<td><code>web.page.regex(host,&lt;path&gt;,&lt;port&gt;,&lt;regex&gt;,&lt;length&gt;,&lt;output&gt;)</code></td>
<td></td>
<td><code>host - hostname or URL (as scheme://host:port/path, where only host is mandatory). Allowed URL schemes: http, https solutions. Missing scheme will be treated as http. If URL is specified path and port must be empty. Specifying user name/password when connecting to servers that require authentication, for example: http://user:password@www.example.com is only possible with cURL support. Punycode is supported in hostnames. path - path to HTML document (default is /) port - port number (default is 80 for HTTP) regex - regular expression describing the required pattern length - maximum number of characters to return output - an optional output formatting template. The </code>\0<code>escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an</code>\N<code> (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).</code></td>
<td>This item turns unsupported if the resource specified in <code>host</code> does not exist or is unavailable. <code>host</code> can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled. <code>host</code> can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled. Content extraction using the output parameter takes place on the agent. The output parameter is supported from version 2.2.</td>
</tr>
<tr>
<td><code>zabbix.stats(ip,&lt;port&gt;)</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Return value</td>
<td>Parameters</td>
<td>Comments</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>Return a set of Zabbix server or proxy internal metrics remotely.</td>
<td>JSON object</td>
<td>ip - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1); port - port of server/proxy to be remotely queried (default is 10051)</td>
<td>Note that the stats request will only be accepted from the addresses listed in the <code>StatsAllowedIP server/proxy</code> parameter on the target instance. A selected set of internal metrics is returned by this item. For details, see Remote monitoring of Zabbix stats.</td>
</tr>
<tr>
<td>zabbix.stats[ip],&lt;port&gt;,queue,&lt;from&gt;,&lt;to&gt;]</td>
<td>Return number of monitored items in the queue which are delayed on Zabbix server or proxy remotely.</td>
<td>JSON object</td>
<td>Note that the stats request will only be accepted from the addresses listed in the <code>StatsAllowedIP server/proxy</code> parameter on the target instance.</td>
</tr>
</tbody>
</table>

**Footnotes**

1 A Linux-specific note. Zabbix agent must have read-only access to filesystem /proc. Kernel patches from [www.grsecurity.org](http://www.grsecurity.org) limit access rights of non-privileged users.

2 **vfs.dev.read[], vfs.dev.write[]**: Zabbix agent will terminate "stale" device connections if the item values are not accessed for more than 3 hours. This may happen if a system has devices with dynamically changing paths or if a device gets manually removed. Note also that these items, if using an update interval of 3 hours or more, will always return '0'.

3 **vfs.dev.read[], vfs.dev.write[]**: 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.

4 SSL (HTTPS) is supported only if agent is compiled with cURL support. Otherwise the item will turn unsupported.

**Encoding settings**

To make sure that the acquired data are not corrupted you may specify the correct encoding for processing the check (e.g. 'vfs.file.contents') in the encoding parameter. The list of supported encodings (code page identifiers) may be found in documentation for [libiconv](http://www.gnu.org) (GNU Project) or in Microsoft Windows SDK documentation for “Code Page Identifiers”.

If no encoding is specified in the encoding parameter the following resolution strategies are applied:

- **Standard resolution** - UTF-8 is used in Unix/Linux (default in newer distributions); ANSI with a system-specific extension is used in Windows;
- **BOM analysis** - applicable for items 'vfs.file.contents', 'vfs.file.regexp', 'vfs.file.regmatch'. An attempt is made to determine the correct encoding by using the byte order mark (BOM) at the beginning of the file. If BOM is not present - standard resolution (see above) is applied instead.

**Troubleshooting agent items**
If used with the 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.