1 Agente Zabbix

Visão geral

Itens do tipo *Agente Zabbix* utilizam o Zabbix Agent para coletar os dados.

As coletas podem ser de dois tipos: passiva ou ativa. Quando estiver configurando o item você pode selecionar um dos tipos a seguir:

- *Agente Zabbix* - para coletas passivas
- *Agente Zabbix (ativo)* - para coletas ativas

Chaves suportadas

A tabela a seguir detalha as chaves disponíveis para monitoração através do Zabbix Agent.

Veja também:

- Itens suportados por plataforma
- Chaves de item específicas para Windows

Parâmetros opcionais e obrigatórios

Parâmetros entre os símbolos **< >** são opcionais.

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
agent.hostname			
Nome do agente (hostname).	Texto		Retorna o valor atual da tag **hostname** no arquivo de configuração do agente.
agent.ping			
Verificação de disponibilidade do agente.	Nothing - unavailable 1 - disponível		Use a função **nodata()** em uma trigger para alertar sobre a indisponibilidade do agente no host.
agent.version			
Versão do Zabbix Agent.	Texto		Exemplo de valor retornado: 1.8.2
kernel.maxfiles			
Quantidade máxima de arquivos abertos simultaneamente que é suportada pelo S.O..	Inteiro		
kernel.maxproc			
Quantidade máxima de processos suportada pelo S.O.	Inteiro		
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Monitoração de arquivos de log.	Log	**file** - caminho completo para o arquivo de log **regexp** - expressão regular descrevendo o padrão desejado **encoding** - codificação da página **maxlines** - Quantidade máxima de linhas que o Zabbix Agent irá enviar para o Zabbix Server ou Zabbix Proxy a cada segundo. Este parâmetro sobrescreve o valor definido em 'MaxLinesPerSecond' no zabbix_agentd.conf **mode** - valores possíveis: *all* todos (padrão), *skip* - não processa (ignora) os dados anteriores existentes no log (afeta somente os novos itens criados). **output** - parâmetro opcional de formatação. A sequência de escape **\0** será substituída pelo texto encontrado, enquanto a sequência de escape **\N** (onde N=1...9) será substituída com o Nézimo grupo compatível (ou um texto vazio caso N exceda a quantidade de grupos compatíveis encontrados).	O item precisa ser configurado como verificação ativa. Se o arquivo não existir ou o usuário da monitoração não possuir privilégios suficientes para acesso, o item ficará como não suportado. Se `output` for deixado em branco: será retornada a linha que corresponder. Observe que qualuqer expressão regular global, exceto a expressão 'Resultado é VERDADEIRO', sempre retornará a linha toda e o parâmetro `output` será ignorado. A extração de conteúdo pelo parâmetro `output` ocorre no lado do agente. Exemplos: ⇒ log[/var/log/syslog] ⇒ log[/var/log/syslog,error] ⇒ log[/home/zabbix/logs/logfile,,,100] O parâmetro `output` é suportado desde o Zabbix 2.2. O parâmetro `mode` é suportado desde o Zabbix 2.0. Veja informações mas detalhadas em: monitoração de log.
logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>]			
Monitoração de log com suporte a rotação de arquivos.	Log	**file_regexp** - caminho absoluto para o arquivo e expressão regular com o padrão desejado **regexp** - expressão regular definindo o conteúdo desejado **encoding** - identificador de código de página **maxlines** - Quantidade máxima de linhas que o Zabbix Agent irá enviar para o Zabbix Server ou Zabbix Proxy a cada segundo. Este parâmetro sobrescreve o valor definido em 'MaxLinesPerSecond' no zabbix_agentd.conf **mode** - valores possíveis: *all* todos (padrão), *skip* - não processa (ignora) os dados anteriores existentes no log (afeta somente os novos itens criados). **output** - parâmetro opcional de formatação. A sequência de escape **\0** será substituída pelo texto encontrado, enquanto a sequência de escape **\N** (onde N=1...9) será substituída com o Nézimo grupo compatível (ou um texto vazio caso N exceda a quantidade de grupos compatíveis encontrados).	O item precisa ser configurado como verificação ativa. A rotação de log é baseada no horário da última modificação dos arquivos. Se `output` for deixado em branco: será retornada a linha que corresponder. Observe que qualuqer expressão regular global, exceto a expressão 'Resultado é VERDADEIRO', sempre retornará a linha toda e o parâmetro `output` será ignorado. A extração de conteúdo pelo parâmetro `output` ocorre no lado do agente. Exemplos: ⇒ logrt["/home/zabbix/logs/^logfile[0-9]{1,3}$",,,100] → irá considerar arquivos que contenham "logfile1" (não irá considerar um arquivo terminando com ".logfile1" somente) ⇒ logrt["/home/user/^logfile_.*_[0-9]{1,3}$","pattern_to_match","UTF-8",100] → coletará a partir de arquivos como "logfile_abc_1" ou "logfile__001". O parâmetro `output` é suportado desde o Zabbix 2.2. O parâmetro `mode` é suportado desde o Zabbix 2.0. Veja informações mas detalhadas em: monitoração de log.
net.dns[<ip>,zone,<type>,<timeout>,<count>,<protocol>]			
Verifica se o serviço do DNS está rodando.	0 - DNS fora do ar (o servidor não respondeu ou a resolução de nome falhou) 1 - DNS está funcional	**ip** - endereço IP do servidor de DNS (deixe em branco para o servidor de DNS padrão, ignorado no Windows) **zone** - zona de DNS a ser testada **type** - tipo do registro a ser consultado (o padrão é *SOA*) **timeout** (ignorado no Windows) - tempo máximo para conclusao da consulta (o padrão é de 1 segundo) **count** (ignorado no Windows) - quantidade de tentativas (padrão é 2) **protocol** - o protocolo utilizado para as consultas de DNS pode ser *udp* (padrão) ou *tcp*	Exemplo: ⇒ net.dns[8.8.8.8,zabbix.com,MX,2,1] Os valores possíveis para o `type` são: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (exceto no Windows), *HINFO*, *MINFO*, *TXT*, *SRV* Nome de domínios internacionalizados não são suportados, favor usar a codificação de nomes IDNA. O parâmetro `protocol` é suportado desde o Zabbix 3.0. O tipo de registro SRV é suportado desde a versão 1.8.6 (no Unix) e 2.0.0 (no Windows). O nome de chave anterior ao Zabbix 2.0 (continua sendo suportado): *net.tcp.dns*
net.dns.record[<ip>,zone,<type>,<timeout>,<count>,<protocol>]			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Executa uma consulta ao DNS.	Sequência de caracteres com o tipo requerido de informação	**ip** - Endereço IP do servidor de DNS (deixe em branco para o servidor de DNS padrão, ignorado no Windows)**zone** - zona de DNS a ser testada **type** - tipo do registro a ser consultado (o padrão é *SOA*) **timeout** (ignorado no Windows) - tempo máximo para conclusao da consulta (o padrão é de 1 segundo) **count** (ignorado no Windows) - quantidade de tentativas (padrão é 2) **protocol** - o protocolo utilizado para as consultas de DNS pode ser *udp* (padrão) ou *tcp*	Exemplo: ⇒ net.dns[8.8.8.8,zabbix.com,MX,2,1] Os valores possíveis para o `type` são: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (exceto no Windows), *HINFO*, *MINFO*, *TXT*, *SRV* Nome de domínios internacionalizados não são suportados, favor usar a codificação de nomes IDNA. O parâmetro `protocol` é suportado desde o Zabbix 3.0. O tipo de registro SRV é suportado desde a versão 1.8.6 (no Unix) e 2.0.0 (no Windows). O nome de chave anterior ao Zabbix 2.0 (continua sendo suportado): *net.tcp.dns.query*
net.if.collisions[if]			
Quantidade de colisões fora da janela.	Inteiro	**if** - nome da interface de rede	
net.if.discovery			
Lista das interfaces de rede. Utilizado no processo de autobusca.	Objeto JSON		Suportado desde a versão 2.0 do Zabbix. Em FreeBSD, OpenBSD e NetBSD suportado desde o Zabbix 2.2. Algumas versões do Windows (por exemplo o Server 2008) poderão necessitar dos últimos updates para suportar caracteres non-ASCII no nome das interfaces.
net.if.in[if,<mode>]			
Estatísticas de entrada na interface de rede.	Inteiro	**if** - nome da interface (Unix); descrição completa da interface de rede ou endereço IPv4 (Windows) **mode** - valores possíveis: *bytes* - quantidade de bytes (padrão) *packets* - quantidade de pacotes *errors* - quantidade de erros *dropped* - quantidade de pacotes 'dropados'	Em ambiente Windows, o item recuperará valores a partir de contadores de 64-bits, se disponíveis. A interface de estatísticas de 64-bit foi introduzida no Windows Vista e Windows Server 2008. Se os contadores de 64-bits não estiverem disponíveis, o agente utilizará contadores de 32-bits. \\Nomes de interfaces com suporte a 'Multi-byte' no windows desde o Zabbix 1.8.6. Exemplos: ⇒ net.if.in[eth0,errors] ⇒ net.if.in[eth0] Você pode obter descrições de interfaces de rede tanto com o item 'net.if.discovery' quanto com item 'net.if.list'. Você pode utilizar esta chave com o recurso de *Delta (alterações por segundo)* para guardar valores com a estatística de bytes por segundo.
net.if.out[if,<mode>]			
Estatísticas de saída na interface de rede.	Inteiro	**if** - nome da interface (Unix); descrição completa da interface de rede ou endereço IPv4 (Windows) **mode** - valores possíveis: *bytes* - quantidade de bytes (padrão) *packets* - quantidade de pacotes *errors* - quantidade de erros *dropped* - quantidade de pacotes 'dropados'	Em ambiente Windows, o item recuperará valores a partir de contadores de 64-bits, se disponíveis. A interface de estatísticas de 64-bit foi introduzida no Windows Vista e Windows Server 2008. Se os contadores de 64-bits não estiverem disponíveis, o agente utilizará contadores de 32-bits. \\Nomes de interfaces com suporte a 'Multi-byte' no windows desde o Zabbix 1.8.6. Exemplos: ⇒ net.if.out[eth0,errors] ⇒ net.if.out[eth0] Você pode obter descrições de interfaces de rede tanto com o item 'net.if.discovery' quanto com item 'net.if.list'. Você pode utilizar esta chave com o recurso de *Delta (alterações por segundo)* para guardar valores com a estatística de bytes por segundo.
net.if.total[if,<mode>]			
Sumarização do tráfego de entrada e de saída na interface de rede.	Inteiro	**if** - nome da interface (Unix); descrição completa da interface de rede ou endereço IPv4 (Windows) **mode** - valores possíveis: *bytes* - quantidade de bytes (padrão) *packets* - quantidade de pacotes *errors* - quantidade de erros *dropped* - quantidade de pacotes 'dropados'	
net.tcp.listen[port]			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Verifica se uma porta TCP está em modo de ESCUTA.	0 - se ela não está em modo de ESCUTA 1 - se estiver em modo de ESCUTA	**port** - Número da porta TCP	Exemplo: ⇒ net.tcp.listen[80] É suportada no Linux desde o Zabbix 1.8.4 A partir do Zabbix 3.0.0, com kernel 2.6.14 ou superior, a informação sobre portas TCP escutando é obtida, se possível, através da interface NETLINK do kernel. Caso contrário ela será obtida nos arquivos /proc/net/tcp e /proc/net/tcp6.
net.tcp.port[<ip>,port]			
Verifica se é possível estabelecer uma conexão TCP com uma porta específica.	0 - não é possível conectar 1 - é possível conectar	**ip** - Endereço IP (o padrão é 127.0.0.1) **port** - número da porta	Exemplo: ⇒ net.tcp.port[,80] → pode ser usado para verificar a disponibilidade de um servidor web rodando na porta 80. Para simples testes de performance TCP, recomenda-se o uso de net.tcp.service.perf[tcp,<ip>,<port>] Observe que estas verificações podem resultar em mensagens adicionais nos arquivos de log dos daemons (sessões SMTP e SSH normalmente são registradas). Nome antigo: *check_port[*]*
net.tcp.service[service,<ip>,<port>]			
Verifica se o serviço esta rodando e aceitando conexões TCP.	0 - serviço fora do ar 1 - serviço em execução	**service** - qualquer um destes: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (veja detalhes) **ip** - Endereço IP (o padrão é 127.0.0.1 **port** - número da porta (na ausência a porta padrão do serviço será usada)	Exemplo: ⇒ net.tcp.service[ftp,,45] → pode ser usado para testar a disponibilidade de um serviço FTP na porta 45. Observe que estas verificações podem resultar em mensagens adicionais nos arquivos de log dos daemons A verificação de protocolos criptografados (tal qual o IMAP na porta 993 ou POP na porta 995) atualmente não é suportada. Uma opção pode ser usar o net.tcp.port para verificaçõs similares a esta. A verificação de LDAP e HTTPS no windows não é suportada. Observe que uma verificação de telnet busca um prompt e logn (':' no final). Osserviços *https* e *telnet* são suportados desde o Zabbix 2.0. Nomenclatura antiga:Nome anterior: *check_service[*]*
net.tcp.service.perf[service,<ip>,<port>]			
Verifica a performance de um serviço TCP.	0 - serviço fora do ar segundos - quantidade de segundos utilizados durante o teste	**service** - um destes: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (veja para maiores detalhes) **ip** - endereço IP (o padrão é 127.0.0.1) **port** - número da porta (se ausente a porta padrão do serviço será utilizada)	Exemplo: ⇒ net.tcp.service.perf[ssh] → pode ser utilizada para testar o tempo de resposta de um servidor SSH. O teste com protocolos criptografados (tal qual IMAP na porta 993 ou POP na porta 995) não é suportado atualmente. Como uma solução alternativa, por favor utilize net.tcp.service.perf[tcp,<ip>,<port>] para verificações como estas. A verificação de servidores LDAP e HTTPS não é suportada atualmente pelo agente no Windows. Observe que um teste de “telnet” espera ter uma tela de login (':' ao final). Os serviços *https* e *telnet* são suportados desde o Zabbix 2.0. Nome antigo: *check_service_perf[*]*
net.udp.listen[port]			
Verifica se uma porta UDP está em modo de escuta.	0 - não está em modo de ESCUTA 1 - está em modo de ESCUTA	**port** - número da porta UDP	Exemplo: ⇒ net.udp.listen[68] É suportado no Linux desde a versão 1.8.4
net.udp.service[service,<ip>,<port>]			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Verifica se um serviço está em execução e respondendo a requisições UDP.	0 - serviço fora do ar 1 - serviço em execução	**service** - *ntp* (veja detalhes) **ip** - endereço IP (o padrão é 127.0.0.1) **port** - número da porta (se ausente a porta padrão do serviço será utilizada)	Exemplo: ⇒ net.udp.service[ntp,,45] → pode ser utilizado para testar a disponibilidade do serviço NTP na porta UDP 45. Este item é suportado desde o Zabbix 3.0.0, mas o serviço *ntp* está disponível através do item net.tcp.service[] nas versões anteriores.
net.udp.service.perf[service,<ip>,<port>]			
Verifica a disponibilidade de um serviço UDP.	0 - serviço fora do ar segundos - quantidade de segundos utilizados durante o teste de conexão	**service** - *ntp* (veja mais detalhes) **ip** - endereço IP (o padrão é 127.0.0.1) **port** - número da porta (se ausente a porta padrão do serviço será utilizada)	Exemplo: ⇒ net.udp.service.perf[ntp] → pode ser utilizado para testar o tempo de resposta de um serviço NTP. Este item é suportado desde o Zabbix 3.0.0, mas o serviço *ntp* está disponível através do item net.tcp.service[] nas versões anteriores.
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]			
Percentual de utilização da CPU.	Numérico	**name** - nome do processo (todos por padrão) **user** - nome do usuário (todos por padrão) **type** - Tipo de utilização de CPU: *total* (padrão), *user*, *system* **cmdline** - filtro por linha de comando (através de expressão regular) **mode** - modo de aquisição de dados: *avg1* (padrão), *avg5*, *avg15* **zone** - zona alvo: *current* (padrão), *all*. Este parâmetro só é suportado na plataforma Solaris.	Exemplos: ⇒ proc.cpu.util[,root] → utilização de CPU de todos os processos rodando sob o usuário “root” ⇒ proc.cpu.util[zabbix_server,zabbix] → UTilização de CPU de todos os processos rodando sob o usuário “zabbix” O valor retornado é baseado em utilização de um núcleo de CPU. Por exemplo, se a utilização de um processo ocupar totalmente dois núcleos, o retorno será igual a 200%. O coletor de utilização de CPU suporta a coleta de, no máximo, 1024 processos únicos (por nome, usuário e linha de comando). Consultas não acessadas nas últimas 24 horas serão removidas. Este item é suportado desde o Zabbix 3.0.0 e está disponível em várias plataformas (veja itens suportados por plataforma).
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]			
Memória utilizado por um processo, em bytes.	Inteiro	**name** - nome do processo (por padrão todos) **user** - nome do usuário (por padrão todos) **mode** - valores possíveis: *avg*, *max*, *min*, *sum* (padrão) **cmdline** - filtro por linha de comando (usando expressão regular) **memtype** - tipo da memória utilizada por processo.	Exemplos: ⇒ proc.mem[,root] → memória utilizada por todos os processos sendo executados sob o usuário “root” ⇒ proc.mem[zabbix_server,zabbix] → memória utilizada por todos os processos 'zabbix_server' que estão sendo utilizados pelo usuuário zabbix ⇒ proc.mem[,oracle,max,oracleZABBIX] → memória utilizada pelos processos de maior consumo rodando no usuário “oracle” e que possuam “oracleZABBIX” em sua linha de comando Veja detalhes sobre como selecionar processos pelos parâmetros `name` e `cmdline`. O parâmetro `memtype` é suportado desde o Zabbix 3.0.0. É suportado em diversas plataformas (veja itens suportados por plataforma). Veja também as notas sobre o parâmetro `memtype`.
proc.num[<name>,<user>,<state>,<cmdline>]			
Número de processos.	Inteiro	**name** - nome do processo (o padrão são todos) **user** - nome do usuário (o padrão são todos) **state** - valores possíveis: *all* (padrão), *run*, *sleep*, *zomb* **cmdline** - filtro por linha de comando (usando expressão regular)	Exemplos: ⇒ proc.num[,mysql] → quantidade de processos rodando com o usuário “mysql” ⇒ proc.num[apache2,www-data] → quantidade de processos “apache2” rodando com o usuário “www-data” ⇒ proc.num[,oracle,sleep,oracleZABBIX] → quantidade de processos em modo “dormente” rodando com o usuário “oracle” que possuem “oracleZABBIX” em sua linha de comando Veja notas sobre seleção de processos com os parâmetros `name` e `cmdline`. No Windows, apenas os parâmetros `name` e `user` são suportados.
sensor[device,sensor,<mode>]			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Sensor de leitura de hardware.	Numérico	**device** - nome do dispositivo **sensor** - nome do sensor **mode** - valores possíveis: *avg*, *max*, *min* (se o parâmetro for omitido, o dispositivo e o sensor serão tratados textualmente).	Lê o arquivo /proc/sys/dev/sensors no Linux 2.4. Exemplo: ⇒ sensor[w83781d-i2c-0-2d,temp1] Antes do Zabbix 1.8.4, o formato *sensor[temp1]* era utilizado.
			Lê o arquivo /sys/class/hwmon no Linux 2.6+. Mais detalhes em sensor para Linux.
			Lê a MIB *hw.sensors* MIB no OpenBSD. Exemplos: ⇒ sensor[cpu0,temp0] → temperatura de uma CPU ⇒ sensor["cpu[0-2]$",temp,avg] → média de temperatura das primeiras três CPU's Suportado no OpenBSD desde o Zabbix 1.8.4.
system.boottime			
Tempo em carga do sistema.	Inteiro (horário em padrão 'Unix timestamp')		
system.cpu.discovery			
Lista de CPUs/Núcleos de CPU detectados. Utilizado pelo processo de autobusca.	Objeto JSON		Suportado em todas as plataformas desde o Zabbix 2.4.0.
system.cpu.intr			
Interrupções do dispositivo.	Inteiro		
system.cpu.load[<cpu>,<mode>]			
Carga de CPU.	Numérico	**cpu** - valores possíveis: *all* (padrão), *percpu* (total dividido pela quantidade de CPUs online) **mode** - valores possíveis: *avg1* (média de um minuto, é o padrão), *avg5*, *avg15*	Exemplo: ⇒ system.cpu.load[,avg5] *percpu* é suportado desde o Zabbix 2.0.0. Antigo nome: *system.cpu.loadX*
system.cpu.num[<type>]			
Quantidade de CPUs.	Inteiro	**type** - valores possíveis: *online* (padrão), *max*	Exemplo: ⇒ system.cpu.num
system.cpu.switches			
Quantidade de mudanças de contexto.	Inteiro		Nome antigo: *system[switches]*
system.cpu.util[<cpu>,<type>,<mode>]			
Utilização percentual de CPU.	Numérico	**cpu** - <*número da CPU*> ou *all* (padrão) **type** - valores possíveis: *idle*, *nice*, *user* (padrão), *system* (padrão para Windows), *iowait*, *interrupt*, *softirq*, *steal*, *guest* (em Linux com kernel 2.6.24 ou superior), *guest_nice* (em Linux com kernel 2.6.33 ou superior) **mode** - valores possíveis: *avg1* (média de um minuto, padrão), *avg5*, *avg15*	Exemplo: ⇒ system.cpu.util[0,user,avg5] Antigo nome: *system.cpu.idleX, system.cpu.niceX, system.cpu.systemX, system.cpu.userX*
system.hostname[<type>]			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Nome do host no sistema operacional.	String	**type** (Apenas para windows, não deve ser utilizado em outros sistemas) - valores possíveis: *netbios* (padrão) ou *host*	O valor é obtido a partir da função GetComputerName() (para **netbios**) ou gethostname() (para **host**) no Windows e pelo comando "hostname" em outros ambientes. Exemplos de retornos possíveis: *no Linux*: ⇒ system.hostname → linux-w7x1 ⇒ system.hostname → www.zabbix.com *no Windows*: ⇒ system.hostname → WIN-SERV2008-I6 ⇒ system.hostname[host] → Win-Serv2008-I6LonG O parâmetro `type` para este item é suportado desde o Zabbix **1.8.6**. Veja o manual de configurações para maiores detalhes.
system.hw.chassis[<info>]			
Informações do Chassis.	String	**info** - valores possíveis: *full* (padrão), *model*, *serial*, *type* ou *vendor*	Exemplo: ⇒ system.hw.chassis[full] → Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXX Desktop] Esta chave depende da disponibilidade da tabela de SMBIOS em memória. **Privilégio de Root** é necessário pois estes dados vem diretamente de leitura de memória. Suportado desde o Zabbix 2.0.
system.hw.cpu[<cpu>,<info>]			
Informações de CPU.	String ou Inteiro	**cpu** - <*Número da CPU*> ou *all* (padrão) **info** - valores possíveis: *full* (padrão), *curfreq*, *maxfreq*, *model* ou *vendor*	Exemplo: ⇒ system.hw.cpu[0,vendor] → AuthenticAMD Recupera informações dos arquivos /proc/cpuinfo e /sys/devices/system/cpu/[cpunum]/cpufreq/cpuinfo_max_freq. Se o número da CPU e *curfreq* ou *maxfreq* for especificado, o retorno será numérico (Hz). Suportado desde o Zabbix 2.0.
system.hw.devices[<type>]			
Lista de dispositivos PCI ou USB.	Text	**type** - *pci* (padrão) ou *usb*	Exemplo: ⇒ system.hw.devices[pci] → 00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge [..] Retorna o texto gerado pelo utilitário lspci ou lsusb (executado sem nenhum parâmetro) Suportado desde o Zabbix 2.0.
system.hw.macaddr[<interface>,<format>]			
Lista de endereços MAC.	String	**interface** - *all* (padrão) ou expressão regular **format** - *full* (padrão) ou *short*	Lista os endereços MAC das interfaces que forem compatíveis com a expressão regular `interface` (*all* lista todas as interfaces). Exemplo: ⇒ system.hw.macaddr["eth0$",full] → [eth0] 00:11:22:33:44:55 Se o `format` for especificado como *short*, os nomes de interface e endereços MAC não serão listados. Suportado desde o Zabbix 2.0.
system.localtime[<type>]			
Horário local.	Inteiro - com `type` como *utc* String - com `type` como *local*	**type** - valores possíveis: *utc* - (padrão) o horário desde Epoch (00:00:00 UTC, 1 de Janeiro de 1970), medido em segundos. *local* - o horário no formato 'yyyy-mm-dd,hh:mm:ss.nnn,+hh:mm'	Parâmetros para este item são suportados desde o Zabbix 2.0.
system.run[command,<mode>]			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Executa um comando no host monitorado.	Retorna o texto gerado pela execução do comando 1 - com o mode como *nowait* (independente do resultado do comando)	**command** - comando a ser executado **mode** - valores possíveis: *wait* - aguarda o fim da execução (padrão), *nowait* - não aguarda	Até 512KB de dados podem ser retornados, incluindo espaços em branco a direita que serão truncados. Para ser processado corretamente, a saída do comando precisa ser um texto. Exemplo: ⇒ system.run[ls -l /] → lista dos arquivos do diretório root. *Nota*: Para habilitar esta funcionalidade, o arquivo de configuração do agente precisa conter a opção EnableRemoteCommands=1. *Nota*: O retorno do item será o texto gerado pela execução normal do comando. *Nota:* Valores vazios são suportados desde o Zabbix 2.4.0. Consulte também: Execução de comandos.
system.stat[resource,<type>]			
Estatísticas do sistema.	Inteiro ou Numérico	**ent** - número de unidades de processador que esta partição tem direito de receber (Numérico) **kthr,<type>** - informação sobre o estado de 'threads' do kernel: *r* - média de 'threads' em execução no kernel (Numérico) *b* - média de 'threads' do kernel na fila do gerenciamento de memória virtual (Numérico) **memory,<type>** - informação sobre utilização de memória real e virtual: *avm* - páginas virtuais ativas (Inteiro) *fre* - tamanho da última lista livre (Inteiro) **page,<type>** - informação sobre atividade e falhas de página: *fi* - entradas de página por segundo (Numérico) *fo* - estatísticas de saidas de página por segundo (Numérico) *pi* - páginas usadas no espaço de paginação space (Numérico) *po* - páginas fora do espaço de paginação (Numérico) *fr* - páginas liberadas (páginas substituídas) (Numérico) *sr* - páginas escaneadas pelo algorítmo de substituição de páginas (Numérico) **faults,<type>** - alerta e taxa de interrupção: *in* - interrupções do dispositivo (Numérico) *sy* - chamadas de sistema (Numérico) *cs* - mudanças de contexto nas 'threads' do kernel (Numérico) **cpu,<type>** - divisão do percentual de tempo do processador: *us* - usuário (Numérico) *sy* - sistema (Numérico) *id* - inativo (Numérico) *wa* - inativo por espera de requisições de disco, NFS ou I/O (Numérico) *pc* - quantidade física de processadores utilizada (Numérico) *ec* - percentual consumido da entidade (Numérico) *lbusy* - indica o percentual de utilização lógica que ocorreu durante a execução em nível de usuário e de sistema (Numérico) *app* - indica a quantidade de processadores disponíveis no 'pool' compartilhado (Numérico) **disk,<type>** - estatísticas de disco: *bps* - indica a quantidade de dado transferido (leitura ou gravação) para a unidade em bytes por segundo (Inteiro) *tps* - indica a quantidade de transferências por segundo que foram solicitadas para o disco/fita (Numérico) Suportado desde o Zabbix 1.8.1.	
system.sw.arch			
Informações da arquitetura de software.	String		Exemplo: ⇒ system.sw.arch → i686 Informação obtida da função 'uname()' . Suportado desde o Zabbix 2.0.
system.sw.os[<info>]			
Informações do sistema operacional.	String	**info** - valores possíveis: *full* (padrão), *short* ou *name*	Exemplo: ⇒ system.sw.os[short]→ Ubuntu 2.6.35-28.50-generic 2.6.35.11 Informação obtida a partir dos arquivos (observe que nem todos os arquivos estão disponíveis em todas as distribuições): /proc/version (*full*) /proc/version_signature (*short*) /etc/issue.net (*name*) Suportado desde o Zabbix 2.0.
system.sw.packages[<package>,<manager>,<format>]			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Lista dos pacotes instalados.	Text	**package** - *all* (padrão) ou expressão regular **manager** - *all* (padrão) ou o gerenciador de pacotes **format** - *full* (padrão) ou *short*	Lista (ordenada alfabeticamente) os pacotes instalados que correspondam à expressão regular `package` (*all* lista todos). Exemplo: ⇒ system.sw.packages[mini,dpkg,short] → python-minimal, python2.6-minimal, ubuntu-minimal Gerenciadores de pacotes suportados (comando executado): dpkg (dpkg --get-selections) pkgtool (ls /var/log/packages) rpm (rpm -qa) pacman (pacman -Q) Se o parâmetro `format` for definido como *full*, os pacotes serão agrupados pelos gerenciadores de pacotes (cada gerenciador em uma linha com seu nome entre colchetes). Se o parâmetro `format` for definido como *short*, os pacotes não serão agrupados e serão presentados em uma linha simples. Suportado desde o Zabbix 2.0.
system.swap.in[<device>,<type>]			
Estatísticas de entrada no Swap (do dispositivo para a memória).	Inteiro	**device** - dispositivo utilizado para o 'swapping' (padrão é *all*) **type** - valores possíveis: *count* (quantidade de 'swapins'), *sectors* (setores armazenados), *pages* (páginas armazenadas). Veja a lista de suporte por plataforma para maiores detalhes.	Exemplo: ⇒ system.swap.in[,pages] A fonte de informações são os arquivos: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
system.swap.out[<device>,<type>]			
Estatísticas de saída do Swap (da memória para o dispositivo).	Inteiro	**device** - dispositivo utilizado para o 'swapping' (o padrão é *all*) **type** - valores possíveis: *count* (quantidade de 'swapouts'), *sectors* (setores armazenados), *pages* (páginas armazenadas). Veja a lista de suporte por plataforma para maiores detalhes.	Exemplo: ⇒ system.swap.out[,pages] A fonte de informações são os arquivos: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
system.swap.size[<device>,<type>]			
Tamanho do Swap em bytes ou em percentual do total.	Inteiro - para bytes Numérico - para percentual	**device** - dispositivo utilizado para o 'swapping' (o padrão é *all*) **type** - valores possíveis: *free* (espaço livre no swap, padrão), *pfree* (percentual de espaço livre no swap), *pused* (percentual de espaço ocupado no swap), *total* (espaço total do swap), *used* (espaço utilizado no swap)	Exemplo: ⇒ system.swap.size[,pfree] → percentual de espaço livre no swap Observe que esta chave pode retornar percentual incorreto em ambientes virtualizados (VMware ESXi, VirtualBox) nas plataformas Windows. Neste caso, recomenda-se o uso da chave 'perf_counter[\700(_Total)\702]' para obter a informação correta do swap. Antigo nome: *system.swap.free*, *system.swap.total*
system.uname			
Informação detalhada do host.	String		Exemplo de valor retornado (Unix): FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386 Exemplo de valor retornado (Windows): Windows ZABBIX-WIN 6.0.6001 Microsoft Windows Server 2008 Standard Service Pack 1 x86 Desde o Zabbix 2.2.0, o valor para este item é obtido com a chamada de sistema 'uname()', anteriormente era obtido com o comando “uname -a” em ambientes Unix. Assim, o valor deste item poderá ser diferente do resultado do comando “uname -a” e não incluirá as informações adicionais presentes no mesmo. No Windows, desde o Zabbix 3.0.0, o valor deste item é obtido a partir das classes WMI 'Win32_OperatingSystem' e 'Win32_Processor'. Anteriormente, ele era obtido através de APIs Windows e chaves não documentadas da registry. O nome do S.O. (incluindo a edição) podem ser traduzidos pelo idioma de apresentação do usuário. Em algumas versões do Windows, pode conter símbolos e espaços adicionais. Observe que no Windows o item retorna a arquitetura do S.O., enquanto no Unix retorna a arquitetura da CPU.
system.uptime			
Tempo de carga em segundos.	Inteiro		Conforme o configuração de itens, use as unidades **s** ou **uptime** para obter valores “legíveis”.
system.users.num			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Quantidade de usuários conectados.	Inteiro		o comando **who** é utilizado no lado do agente para obter a resposta.
vfs.dev.read[<device>,<type>,<mode>]			
Estatísticas de leitura de disco.	Inteiro - com `type` em *sectors*, *operations*, *bytes* Numérico - com `type` em *sps*, *ops*, *bps*	**device** - dispositivo de disco (padrão é *all*) **type** - valores possíveis: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* Este parâmetro precisa ser especificado, uma vez que os padrões diferem entre os SOs. *sps*, *ops*, *bps* unidades: setores, operações, bytes por segundo, respectivamente. **mode** - valores possíveis: *avg1* (média de um minuto, padrão), *avg5*, *avg15*. Este parâmetro é suportado apenas com `type` definido como: sps, ops, bps.	O valor padrão do parâmetro 'type' em diferentes SOs: AIX - operations FreeBSD - bps Linux - sps OpenBSD - operations Solaris - bytes Exemplo: ⇒ vfs.dev.read[,operations] *sps*, *ops* and *bps* em plataformas suportadas utilizado para ser limitado a 8 dispositivos (7 individuais e um global). Desde o Zabbix 2.0.1 este limite foi elevado para 1024 dispositivos (1023 individuais e um global). Se o primeiro parâmetro for definido como *all* a chave retornará um resumo das estatísticas, incluindo todos os dispositivos de bloco tal qual: sda, sbd e suas partições (sda1, sda2, sdb3...) alem de múltiplos dispositivos (MD raid) baseados em dispositivos de bloco ou partições e volumes lógicos (LVM). Nestes casos o valor retornado deverá ser considerado apenas como um valor relativo (dinâmico em função do tempo), nunca como absoluto. Suporte a LVM desde o Zabbix 1.8.6. Apenas nomes relativos de disco poderão ser utilizados (por exemplo, **sda**) desde o Zabbix 1.8.6. Desde esta versão, um prefixo opcional **/dev/** pode ser utilizado (por exemplo, **/dev/sda**). Antigo nome: *io[*]*
vfs.dev.write[<device>,<type>,<mode>]			
Estatísticas de gravação no disco.	Inteiro - com `type` em *sectors*, *operations*, *bytes* Numérico - com `type` em *sps*, *ops*, *bps*	**device** - dispositivo de disco (padrão é *all*) **type** - valores possíveis: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* Este parâmetro precisa ser especificado, uma vez que os padrões diferem entre os SOs. *sps*, *ops*, *bps* unidades: setores, operações, bytes por segundo, respectivamente. **mode** - valores possíveis: *avg1* (média de um minuto, padrão), *avg5*, *avg15*. Este parâmetro é suportado apenas com `type` definido como: sps, ops, bps.	O valor padrão do parâmetro 'type' em diferentes SOs: AIX - operations FreeBSD - bps Linux - sps OpenBSD - operations Solaris - bytes Exemplo: ⇒ vfs.dev.write[,operations] *sps*, *ops* and *bps* em plataformas suportadas utilizado para ser limitado a 8 dispositivos (7 individuais e um global). Desde o Zabbix 2.0.1 este limite foi elevado para 1024 dispositivos (1023 individuais e um global). Se o primeiro parâmetro for definido como *all* a chave retornará um resumo das estatísticas, incluindo todos os dispositivos de bloco tal qual: sda, sbd e suas partições (sda1, sda2, sdb3...) alem de múltiplos dispositivos (MD raid) baseados em dispositivos de bloco ou partições e volumes lógicos (LVM). Nestes casos o valor retornado deverá ser considerado apenas como um valor relativo (dinâmico em função do tempo), nunca como absoluto. Suporte a LVM desde o Zabbix 1.8.6. Apenas nomes relativos de disco poderão ser utilizados (por exemplo, **sda**) desde o Zabbix 1.8.6. Desde esta versão, um prefixo opcional **/dev/** pode ser utilizado (por exemplo, **/dev/sda**). Antigo nome: *io[*]*
vfs.file.cksum[file]			
Verificação de arquivo, calculada através do algoritmo 'cksum' do UNIX.	Inteiro	**file** - caminho completo para o arquivo	Exemplo: ⇒ vfs.file.cksum[/etc/passwd] Exemplo de valor retornado: 1938292000 Antigo nome: *cksum* O limite de tamanho de arquivo depende do suporte a grandes arquivos.

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
vfs.file.contents[file,<encoding>]			
Recupera o conteúdo de um arquivo.	Text	**file** - caminho completo para o arquivo **encoding** - identificador de código de página	Retorna uma 'string' vazia se o arquivo estiver vazio ou contitver somente os caractéres LF/CR apenas. Exemplo: ⇒ vfs.file.contents[/etc/passwd] Este item é limitado a arquivos de até 64 Kbytes. Suportado a partir do Zabbix 2.0.
vfs.file.exists[file]			
Verifica se um arquivo existe.	0 - não encontrado 1 - arquivo normal ou um link (simbólico ou 'hard') para um arquivo normal existente	**file** - caminho completo para o arquivo	Exemplo: ⇒ vfs.file.exists[/tmp/application.pid] O valor retornado depende do que a macro S_ISREG POSIX retornar. O tamanho do arquivo depende do suporte a grandes arquivos.
vfs.file.md5sum[file]			
Sumariação MD5 de um arquivo.	String de caracteres ('hash' MD5 de um arquivo)	**file** - caminho completo para o arquivo	Exemplo: ⇒ vfs.file.md5sum[/usr/local/etc/zabbix_agentd.conf] Exemplo de valor retornado: b5052decb577e0fffd622d6ddc017e82 O limite de tamanho de arquivo (64 MB) para este item foi removido no Zabbix 1.8.6. O tamanho do arquivo depende do suporte a grandes arquivos.
vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]			
Procura por um texto em um arquivo.	A linha contendo o texto, ou o especificado no parâmetro opcional `output`	**file** - caminho completo para o arquivo **regexp** - expressão regular no padrão GNU **encoding** - identificador do código de página **start line** - número da primeira linha a pesquisar (por padrão é a primeira linha do arquivo). **end line** - número da última linha a pesquisar (por padrão a última linha do arquivo). **output** - parâmetro opcional de formatação. A sequência de escape **\0** será substituída pelo texto encontrado enquanto a sequência de escape **\N** (onde N=1...9) será substituída pela Nth ocorrência (ou um texto vazio se N for maior que a quantidade de ocorrências).	Apenas a primeira linha encontrada será retornada. Será retornado um texto vazio se não for localizada nenhuma linha compatível com a expressão. A extração de conteúdo utilizando o parâmetro `output` será feita pelo agente. Os parâmetros `start line`, `end line` e `output` são suportados desde o Zabbix 2.2. Exemplos: ⇒ 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] → recuperando o ID do usuário *zabbix*
vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]			
Procura por um texto em um arquivo.	0 - não encontrou 1 - encontrou	**file** - caminho completo para o arquivo **regexp** - expressão regular no padrão GNU **encoding** - identificador de código de página **start line** - número da primeira linha a procurar (por padrão a primeira linha do arquivo). **end line** - número da última linha a procurar (por padrão a última linha do arquivo).	Os parâmetros `start line` e `end line` são suportados desde o Zabbix 2.2. Exemplo: ⇒ vfs.file.regmatch[/var/log/app.log,error]
vfs.file.size[file]			
Tamanho do arquivo (em bytes).	Inteiro	**file** - caminho completo para o arquivo	O arquivo deve permitir a leitura pelo usuário *zabbix*. Exemplo: ⇒ vfs.file.size[/var/log/syslog] O tamanho do arquivo depende do definido no suporte a grandes arquivos.

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
vfs.file.time[file,<mode>]			
Informação de hora do arquivo.	Inteiro (no padrão 'Unix timestamp')	**file** - caminho completo para o arquivo **mode** - valores possíveis: *modify* (padrão) - momento de modificação, *access* - momento do último acesso, *change* - momento da última modificação	Exemplo: ⇒ vfs.file.time[/etc/passwd,modify] O tamanho do arquivo depende do definido no suporte a grandes arquivos.
vfs.fs.discovery			
Lista de sistemas de arquvios montados. Utilizado pelo processo de autobusca.	Objeto JSON		Suportado desde o Zabbix 2.0. A macro {#FSDRIVETYPE} é suportada no Windows desde o Zabbix 3.0.
vfs.fs.inode[fs,<mode>]			
Percentual de inodes.	Inteiro - para número Numérico - para percentual	**fs** - sistema de arquivo **mode** - valores possíveis: *total* (padrão), *free*, *used*, *pfree* (percentual livre), *pused* (percentual em uso)	Exemplo: ⇒ vfs.fs.inode[/,pfree] Antigo nome: *vfs.fs.inode.free[*]*, *vfs.fs.inode.pfree[*]*, *vfs.fs.inode.total[*]*
vfs.fs.size[fs,<mode>]			
Espaço em disco em bytes ou percentual do total.	Inteiro - para bytes Numérico - para percentual	**fs** - sistema de arquivos **mode** - valores possíveis: *total* (padrão), *free*, *used*, *pfree* (percentual livre), *pused* (percentual em uso)	Em caso de um volume montado o espaço total do sistema de arquivo será retornado. Exemplo: ⇒ vfs.fs.size[/tmp,free] O espaço reservado ao sistema de arquivos é levado em conta e não incluído quando é utilizado o modo *free*. Antigo padrão de nome: *vfs.fs.free[*]*, *vfs.fs.total[*]*, *vfs.fs.used[*]*, *vfs.fs.pfree[*]*, *vfs.fs.pused[*]*
vm.memory.size[<mode>]			
Tamanho da memória em bytes ou em percentual.	Inteiro - para bytes Numérico - para percentual	**mode** - valores possíveis: *total* (padrão), *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *wired*, *used*, *pused* (percentual de uso), *available*, *pavailable* (percentual disponível)	Este item suporta três categorias de parâmetros: 1) *total* - total de memória; 2) tipos específicos de memória (dependentes de plataforma): *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *wired*; 3) Estimativa de memória no nível de usuário: *used*, *pused*, *available*, *pavailable*. Veja mais detalhes sobre 'vm.memory.size' em parâmetros de memória. Antigo nome: *vm.memory.buffers*, *vm.memory.cached*, *vm.memory.free*, *vm.memory.shared*, *vm.memory.total*
web.page.get[host,<path>,<port>]			
Recupera o conteúdo de uma página web.	A página web é recuperada como um texto	**host** - nome do host **path** - caminho para o arquivo HTML (o padrão é /) **port** - número da porta (o padrão é 80)	Retorna uma string vazia em caso de falha. Exemplo: ⇒ web.page.get[www.zabbix.com,index.php,80]
web.page.perf[host,<path>,<port>]			
Tempo para carga da página completa (em segundos).	Numérico	**host** - nome do host **path** - caminho para o arquivo HTML (o padrão é /) **port** - número da porta (o padrão é 80)	Retorna 0 caso falhe. Exemplo: ⇒ web.page.perf[www.zabbix.com,index.php,80]
web.page.regexp[host,<path>,<port>,<regexp>,<length>,<output>]			

Chave			
Descrição	**Valor retornado**	**Parâmetros**	**Comentários**
Procura por um texto em uma página web.	O texto localizado, ou o especificado pelo parâmetro opcional `output`	**host** - nome do host **path** - caminho para o arquivo HTML (o padrão é /) **port** - número da porta (o padrão é 80) **regexp** - expressão regular no padrão GNU **length** - quantidade máxima de caracteres a serem retornados **output** - parâmetro opcional de formatação. A sequência de caracteres **\0** será substituída pelo texto encontrado, enquanto a sequência **\N** (onde N=1...9) será substituída pela Nth ocorrência (ou um texto em branco caso N exceda a quantidade de ocorrências).	Retorna um texto em branco caso não seja encontrada uma ocorrência. A extração de conteúdo utilizando o parâmetro `output` será executada no agente. O parâmetro `output` é suportado desde o Zabbix 2.2. Exemplo: ⇒ web.page.regexp[www.zabbix.com,index.php,80,OK,2]

Especificamente para o Linux: O Zabbix Agent precisa ter acesso de leitura ao sistema de arquivos */proc*. Existem patches de kernel em www.grsecurity.org para limitar os privilégios.

Codificações de página disponíveis

O parâmetro `encoding` é utilizado para especificar um código de página para processar as verificações de itens, para que o dado não seja corrompido. Para uma lista de codificações suportadas (identificadores de código de página), favor consultar a documentação relativa, tal qual a documentação para o libiconv (projeto GNU) ou a documentação do Microsoft Windows para "Identificadores de código de página".

Se não for definido o parâmetro `encoding`, será utilizado o UTF-8 (código de página padrão em distribuições Unix/Linux, veja suas configurações do SO) ou ANSI com extensão específica para o ambiente (Windows).

Resolução de problemas com itens de Agente

1. Se for utilizada uma verificação passiva, o parâmetro, *Timeout* no arquivo de configuração do servidor precisa ser maior do que o definido no arquivo de configuração do agente. A inobservância disso poderá impedir que o servior colete o dado.