> 《Zabbix企业级分布式监控系统第2版》随书代码
>
> 代码仓库地址 https://github.com/zabbix-book/zabbix_v2
>
> 书籍购买地址 https://item.jd.com/12653708.html

124页

```
监控项存在于zabbix.items表中：

mysql> select * from zabbix.items limit 1\G;
```

126页

```
shell# zabbix_get  -s  127.0.0.1  -k  vm.memory.size[pavailable]
89.833440
```

131页

```
5.2.3  key的参数数组应用实例
多个参数之间用逗号（,）隔开，每个参数对key分别传递参数的值。

UserParameter=wc[*],grep -c "$2" $1
上述语句表示把$2、$1的值传递给key，测试如下：

shell# zabbix_get  -s   127.0.0.1 -k  wc[/etc/passwd,root]
注意，这里的/etc/passwd为$1，root为$2，则key最终运行的命令为grep -c root
/etc/passwd。
如果方括号“[]”中有多个参数的值，那么各值之间用逗号（,）隔开。例如：

icmpping[,,200,,500]
5.2.4  用户自定义参数
用户自定义参数（UserParameter）仅支持Agent的方式，对于其他方式它是不支持的。
1. key自定义的语法格式
在/etc/zabbix/zabbix_agentd.conf中配置参数，写法如下：

UserParameter=key,command
除了上面这种写法，还支持参数传递的写法，具体如下：

UserParameter=key[*],command  $1  $2  $3  ……
```

```
shell# vim  /etc/zabbix/zabbix_agentd.conf
UnsafeUserParameters=1
前面已经介绍了key名称的
```

132页

```
4. 自定义key的例子
在 /etc/zabbix/zabbix_agentd.conf后面添加如下内容:

UserParameter=get.os.type, head -1  /etc/issue
然后重启zabbix_agentd服务（注意，修改配置后必须重启服务）。

shell# service  zabbix_agentd   restart
运行测试命令，查看key，语句如下:

shell# zabbix_get -s 127.0.0.1 -k get.os.type
CentOS release 6.5 (Final)
```

```
5. 子配置文件的配置
为了便于维护和分类管理，UserParameter的内容可以单独写一个配置文件。

shell# vim  /etc/zabbix/zabbix_agentd.conf
Include=/etc/zabbix/zabbix_agentd.conf.d/
```

133页

```
（1）修改Agent配置文件。
shell# vim  /etc/zabbix/zabbix_agentd.conf
UnsafeUserParameters=1                          #处理特殊字符
Include=/etc/zabbix/zabbix_agentd.conf.d/      #子配置文件路径
（2）修改子配置文件。

shell# vim  /etc/zabbix/zabbix_agentd.conf.d/get_os_type.conf
UserParameter=get.os.type, head -1  /etc/issue
  #自定义key，如有参数传递，请参考前面的内容
（3）重启服务测试key。

shell# service  zabbix-agent   restart          #重启服务
shell# zabbix_get -s 127.0.0.1 -k get.os.type     #测试key获取参数
CentOS release 6.4 (Final)                      #key获取的值
（4）用zabbix_agentd查看key是否被支持。
```

```
shell# zabbix_agentd -p|grep get\.os
get.os.type                                     [t|CentOS release 6.5 (Final)]
```

如果能看到key名称，且能看到获取到的数据，则说明自定义的key是正确的。

（5）在Web页面添加Item，注意数据类型的选择。

135页

用zabbix_get测试数据获取情况（关于zabbix_get的用法请参考3.7节）：

```
[root@www ~]# zabbix_get  -s  127.0.0.1  -k  net.if.in[eth0,bytes]
358589160
```

136-138页

如果想获取网卡接收的数据包数量，用net.if.in[eth0,packets]即可。

```
[root@www ~]# zabbix_get  -s  127.0.0.1  -k  net.if.in[eth0,packets]
257021
```

这里的packets其实就是在ifconfig命令中看到的packets。

```
[root@www ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:12:F6:05
          inetaddr:192.168.1.9 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe12:f605/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:257021 errors:0 dropped:0 overruns:0 frame:0
          TX packets:165997 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:357444550 (340.8 MiB)  TX bytes:11420540 (10.8 MiB)
```

另外，关于网卡流量监控的key如下（具体用法见官方文档解释）：

```
net.if.out[if,<mode>]
net.if.collisions[if]
net.if.discovery
net.if.out[if,<mode>]
net.if.total[if,<mode>]
```

2. 与网络连接相关的key

```
net.tcp.listen[port]
net.tcp.port[<ip>,port]
net.tcp.service[service,<ip>,<port>]
```

```
net.tcp.service.perf[service,<ip>,<port>]
net.udp.listen[port]
net.udp.service[service,<ip>,<port>]
net.udp.service.perf[service,<ip>,<port>]
net.dns[<ip>,name,<type>,<timeout>,<count>,<protocol>]
net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]
```

3. 监控进程的key

```
kernel.maxfiles
kernel.maxproc
proc.mem[<name>,<user>,<mode>,<cmdline>]
proc.num[<name>,<user>,<state>,<cmdline>]
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]
```

4. 监控CPU和内存的key

```
system.cpu.intr
system.cpu.load[<cpu>,<mode>]
system.cpu.num[<type>]
system.cpu.switches
system.cpu.util[<cpu>,<type>,<mode>]
    system.cpu.discovery
vm.memory.size[<mode>]
system.swap.in[<device>,<type>]
system.swap.out[<device>,<type>]
system.swap.size[<device>,<type>]
sensor[device,sensor,<mode>]
```

5. 磁盘I/O监控的key

```
vfs.dev.read[<device>,<type>,<mode>]
vfs.dev.write[<device>,<type>,<mode>]
vfs.fs.inode[fs,<mode>]
```

6. 文件监控的key

```
vfs.file.cksum[file]
vfs.file.contents[file,<encoding>]
vfs.file.exists[file]
vfs.file.md5sum[file]
vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]
vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]
vfs.file.size[file]
vfs.file.time[file,<mode>]
Vfs.fs.discovery
vfs.fs.size[fs,<mode>]
vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,
<max_depth>,<min_size>,<max_size>,<min_age>,<max_age>]
```

7. 与操作系统相关的key

```
system.boottime
system.hw.chassis[<info>]
```

```
system.hw.cpu[<cpu>,<info>]
system.hw.devices[<type>]
system.hw.macaddr[<interface>,<format>]
system.localtime[<type>]
system.run[command,<mode>]
system.stat[resource,<type>]
system.sw.arch
system.sw.os[<info>]
system.sw.packages[<package>,<manager>,<format>]
system.uname
system.uptime
system.users.num
8. 与Web性能相关的key

web.page.get[host,<path>,<port>]
web.page.perf[host,<path>,<port>]
web.page.regexp[host,<path>,<port>,<regexp>,<length>,<output>]
9. 监控硬件信息的key

sensor[device,sensor,<mode>]
10. 日志监控的key
需要主动模式的支持。

log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]
logrt[file_pattern,<regexp>,<encoding>,<maxlines>,<mode>,<output>]
关于更多的Zabbix-Agent Key，请读者参考官方文档，地址如下：
https://www.zabbix.com/documentation/4.0/manual/config/items/itemtypes/zabbix_a
gent
11. Windows监控的key

eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]
net.if.list
perf_counter[counter,<interval>]
proc_info[<process>,<attribute>,<type>]
service.discovery
service.info[service,<param>]
service_state[*]
services[<type>,<state>,<exclude>]
wmi.get[<namespace>,<query>]
关于key的详细用法，读者可以参考如下地址：
https://www.zabbix.com/documentation/4.0/manual/config/items/itemtypes/zabbix_a
gent/win_keys
```

139页

```
C:\> typeperf -qx > performance_counters.txt  #查看结果如图5-8所示
在Windows中安装好Zabbix-Agent后，配置zabbix_agentd.conf，即可采集数据。

C:\> "C:\Program Files\zabbix_agents_4.0.0.win\in\win64\zabbix_agentd.exe"  -c
 "c:\Program Files\zabbix_agents_4.0.0.win\conf\zabbix_agentd.win.conf" -t
perf_counter["\Processor(_Total)\% Idle Time"]
perf_counter[\Processor(_Total)\% Idle Time]  [d|92.035326]
```

141-143页

```
1. Simple check支持的key

icmpping[<target>,<packets>,<interval>,<size>,<timeout>]
icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]
icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]
net.tcp.service[service,<ip>,<port>]
net.tcp.service.perf[service,<ip>,<port>]
net.udp.service[service,<ip>,<port>]
net.udp.service.perf[service,<ip>,<port>]
下面我们来分别介绍上面的key。
icmpping[<target>,<packets>,<interval>,<size>,<timeout>]
含义：这个key是用来ping目标IP地址是否存活的。
检测结果：0表示不存活，1表示存活。
数据类型：整数。
target：可以不用填写，可以为IP地址，也可以为DNS。
packets：表示ping多少个数据包。
interval：表示间隔多久ping一次，单位为毫秒。
size：表示包的大小，单位是字节。
timeout：表示超时时间，单位为毫秒。
举例：icmpping[,3,10]表示ping 3个包，间隔10毫秒，检测目标IP地址是否存活。
icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]
含义：这个key是用来ping目标IP地址的丢包率的。参数同icmpping。
数据类型：浮点数。
举例：icmppingloss[,3]表示ping 3个包，间隔10毫秒，检测目标IP地址的丢包率。
icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]
含义：这个key是用来ping目标IP地址的平均响应时间的。
数据类型：浮点数。
mode：表示对ping的响应时间的计算，支持avg、max、min。其他参数同icmpping。
举例：icmppingloss[,3,,,,avg]表示ping 3个包，检测目标IP地址的平均响应时间。
net.tcp.service[service,<ip>,<port>]
含义：检测目标TCP服务的存活。
数据类型：整数。
检测结果：0表示不存活，1表示存活。
service：可以为ssh、ldap、smtp、ftp、http、pop、nntp、imap、tcp、https、telnet。
ip：表示连接service的IP地址，默认为Item的接口。
port：表示去连接service的IP地址，默认为service的标准端口。
举例：net.tcp.service[ftp]表示检测FTP是否存活。
```

net.tcp.service.perf[service,<ip>,<port>]
含义：检测目标TCP服务的连接响应时间。参数同net.tcp.service。
数据类型：浮点数。
检测结果：0.000000表示服务不存活。
举例：net.tcp.service.perf[ftp]表示检测FTP的连接响应时间。
net.udp.service[service,<ip>,<port>]
含义：检测目标UDP服务的存活。
数据类型：整数。
检测结果：0表示不存活，1表示存活。
service：可以为ntp。
ip：表示连接service的IP地址，默认为Item的接口。
port：表示连接service的IP地址，默认为service的标准端口。
举例：net.udp.service[ntp]表示检测NTP是否存活。
net.udp.service.perf[service,<ip>,<port>]
含义：检测目标UDP服务的连接响应时间。参数同net.udp.service。
数据类型：浮点数。
检测结果：0.000000表示服务不存活。
举例：net.udp.service.perf[ntp]表示检测NTP的连接响应时间。
关于这些key的详细说明，请参考官方文档，地址为：
https://www.zabbix.com/documentation/4.0/manual/config/items/itemtypes/simple_checks
2. 超时处理
超过zabbix_server.conf中设置的超时时间（最大超时时间为30秒）后，Zabbix会放弃处理。
3. ICMP ping
Zabbix 用fping处理ICMP ping请求，所以需要安装fping程序。在zabbix_server.conf中，FpingLocation参数是用于配置fping程序路径的。
由于fping默认是以root权限工作的，而Zabbix-Server是Zabbix用户运行的，所以需要对fping程序设置setuid（如果在自定义key时需要用到netstat命令，也同样要设置setuid；否则不能获取到数据，而在日志中提示权限拒绝）。

```
shell# chown  root:zabbix   /usr/sbin/fping
shell# chmod  4710   /usr/sbin/fping
```

145页

```
log[/path/to/file/file_name,<regexp>,<encoding>,<maxlines>,<mode>,<output>,
<maxdelay>,<options>]
logrt[/path/to/file/regexp_describing_filename_pattern,<regexp>,<encoding>,
<maxlines>,<mode>,<output>,<maxdelay>,<options>]
log.count[/path/to/file/file_name,<regexp>,<encoding>,<maxproclines>,<mode>,
<maxdelay>,<options>]
logrt.count[/path/to/file/regexp_describing_filename_pattern,<regexp>,
<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>]
```

146页

例如，我们定义了以下的key：

```
log[/path/to/the/file,"large result buffer allocation.*Entries: ([0-9]+)",,,,\1]
```

当日志中出现以下记录时，则匹配，其中([0-9]+)会匹配到Entries后面的数值，将由“\1”输出，该key输出的数据类型为整数。该key匹配输出的是数值，而非整条日志记录。

```
Fr Feb 07 2014 11:07:36.6690 */ Thread Id 1400 (GLEWF) large result buffer allocation - /Length: 437136/Entries: 5948/Client Ver: >=10/RPC ID: 41726453/User: AUser/Form: CFG:ServiceLevelAgreement
```

147页

```
[root@zabbix ~]# tail -f  /var/log/zabbix/zabbix_agentd.log|grep log
 "key":"log[\/var\/log\/secure,session,,50]",22851:20140106:225042.451 In add_check() key:'log[/var/log/secure,session,,50]' refresh:30 lastlogsize:0 mtime:0
 22851:20140106:225042.451 In process_log() filename:'/var/log/secure' lastlogsize:0
 22851:20140106:225042.452 cannot open '/var/log/secure': [13] Permission denied
 22851:20140106:225042.452 active check "log[/var/log/secure,session,,50]" is not supported
 22851:20140106:225042.452 In process_value() key:'Zabbix server:log[/var/log/secure,session,,50]' value:'(null)'
 "key":"log[\/var\/log\/secure,session,,50]",
```

148页

```
 [root@localhost ~]# ls -l /var/log/secure
-rw------- 1 root root 3788 Jul 27 15:04 /var/log/secure
```

在这里，为了演示如何解决这个问题而改变文件的权限，使用如下命令：

```
shell# chown  zabbix.root  /var/log/secure
```

对于不方便设置权限的日志文件，可以使Zabbix-Agent采用root权限运行：

```
shell# vim /etc/zabbix/zabbix_agentd.conf
### Option: AllowRoot
#       Allow the agent to run as 'root'. If disabled and the agent is started by 'root', the agent
```

```
#       will try to switch to the user specified by the User configuration
option instead.
#       Has no effect if started under a regular user.
#       0 - do not allow
#       1 - allow
#
# Mandatory: no
# Default:
# AllowRoot=0
AllowRoot=1

shell# systemctl restart zabbix-agent
```

151页

```
mysql> select * from history_log;
```

```
shell# zabbix_get -s 127.0.0.1 -k log[/var/log/secure,session]
ZBX_NOTSUPPORTED
```

153页

```
2. 计算表达式的例子
（1） 计算剩余磁盘的百分比。

100*last("vfs.fs.size[/,free]",0)/last("vfs.fs.size[/,total]",0)
（2）计算10分钟内Zabbix values的可用大小。

avg("Zabbix server:zabbix[wcache,values]",600)
（3）统计eth0的进出流量总和。

last("net.if.in[eth0,bytes]",0)+last("net.if.out[eth0,bytes]",0)
（4）统计进流量占网卡总流量的百分比。

100*last("net.if.in[eth0,bytes]",0)/(last("net.if.in[eth0,bytes]",0)+last("net.
if.out[eth0,bytes]",0))
（5）对聚合监控方式的监控项（Aggregated[能写成中文吗？] Item）进行计算，注意引号需要转义。

last("grpsum[\"video\",\"net.if.out[eth0,bytes]\",\"last\",\"0\"]",0) /
last("grpsum[\"video\",\"nginx_stat.sh[active]\",\"last\",\"0\"]",0)
```

154页

3. 示例1：计算磁盘总剩余容量
例如要计算C、D、E、F磁盘的总剩余容量，分区剩余容量的key为：

```
vfs.fs.size[C:,free]
vfs.fs.size[D:,free]
vfs.fs.size[E:,free]
vfs.fs.size[F:,free]
```

添加磁盘总剩余容量的计算表达式，如图5-26所示。

```
last("vfs.fs.size[C:,free]",0)+last("vfs.fs.size[D:,free]",0)+last("vfs.fs.size[E:,free]",0)+last("vfs.fs.size[F:,free]",0)
```

155页

4. 示例2：计算网卡的流量
计算表达式如下：

```
last("net.if.in[eth0]",0)+last("net.if.in[eth1]",0)+last("net.if.in[eth2]",0)
```

160-161页

3. 聚合型监控方式配置实例

```
grpsum["MySQL Servers","vfs.fs.size[/,total]",last]
```

含义：对MySQL Servers组监控项vfs.fs.size[/,total]，最近一次监控数据求和，即求该组/分区总的容量。

```
grpavg["MySQL Servers","system.cpu.load[,avg1]",last]
```

含义：对MySQL Servers组监控项system.cpu.load[,avg1]，最近一次监控数据求平均值，可以得出该组的load平均值。

```
grpavg["MySQL Servers",mysql.qps,avg,5m]
```

含义：对MySQL Servers组监控项mysql.qps，最近5分钟内每个主机所获取到的监控数据的平均值（avg），再次求平均值（grpavg），得出该组mysql.qps 5分钟内的平均值。

```
grpavg[["Servers A","Servers B","Servers C"],system.cpu.load,last]
```

含义：对Servers A、Servers B、Servers C三个主机组监控项system.cpu.load，每个主机的最近一次监控数据求平均值。

```
grpsum[["WEB-1"," WEB-2","WEB-3"],nginx.404.log ,sum,10m]
```

含义：对WEB-1、WEB-2、WEB-3三个主机组监控项nginx.404.log，每个主机近10分钟出现404的次数和，再次求和（grpsum），即求三3个分组404一共出现的次数。

```
grpsum["Linux",log[/var/log/message,error],count,30m]
```

含义：对Linux主机组监控项log[/var/log/message,error]，每个主机近30分钟出现的错误日志次数[指什么次数？]（即日志的条数），再次求和（grpsum），即求该组一共出现的匹配错误日志的个数。

164页

2. SSH监控配置
在默认情况下，Zabbix-Server并不知道我们使用哪个SSH密钥来连接服务器，因此需要指定SSH密钥的位置。由于使用RPM包安装的Zabbix-Server，其用户家目录在/var/lib/zabbix目录下面，因此我们将SSH密钥目录设置为/var/lib/zabbix/.ssh。

```
shell#  vim /etc/zabbix/zabbix_server.conf
### Option: SSHKeyLocation
#       Location of public and private keys for SSH checks and actions.
#
# Mandatory: no
# Default:
# SSHKeyLocation=
SSHKeyLocation=/var/lib/zabbix/.ssh

shell# systemctl restart zabbix-server
```

3. 生成SSH密钥
使用如下命令生成SSH密钥：

```
shell# mkdir -p /var/lib/zabbix/.ssh
shell# chown zabbix:zabbix -R /var/lib/zabbix/.ssh
shell# sudo -u zabbix ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/zabbix/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): #直接按回车键
Enter same passphrase again:   #可以不用输入任何参数直接按回车键
Your identification has been saved in /var/lib/zabbix/.ssh/id_rsa
Your public key has been saved in /var/lib/zabbix/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:AvvQqLfE2RsqQtMQKAY7Z9wxP+e/OxglqLY6S9Cde2w zabbix@localhost.localdomain
The key's randomart image is:
+---[RSA 2048]----+
|+   o             |
|o= . +            |
|= = o o..         |
| * . *.+. .       |
|. + *.o So        |
| + +oB ...        |
|. +.*.E  o.       |
|..oo.= o. ..      |
| .o=o .   oo      |
+----[SHA256]-----+
```

```
shell# ls /var/lib/zabbix/.ssh -l
-rw------- 1 zabbix zabbix 1679 Jul 27 17:09 id_rsa
-rw-r--r-- 1 zabbix zabbix  410 Jul 27 17:09 id_rsa.pub
```

4. 分发SSH密钥

现在我们将SSH密钥分发到目标机器10.0.2.43。假如有更多的目标机器，我们需要批量复制到目标机器。

```
[root@localhost ~]# sudo -u zabbix ssh-copy-id root@10.0.2.43
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/var/lib/zabbix/
.ssh/id_rsa.pub"
The authenticity of host '10.0.2.43 (10.0.2.43)' can't be established.
RSA key fingerprint is SHA256:r+AAt7xmf8NvZ1pEHXD97K7BiMUhHAZv0FmxmPAnf1A.
RSA key fingerprint is MD5:8f:ef:ae:ec:61:40:c2:30:cb:cf:47:7c:1b:3a:f8:17.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
root@10.0.2.43's password:   #输入10.0.2.43这台机器的root密码
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'root@10.0.2.43'"
and check to make sure that only the key(s) you wanted were added.
```

测试目标机器能否免密码访问，使用如下命令：

```
[root@localhost ~]# sudo -u zabbix ssh root@10.0.2.43 'ifconfig'
eth0   Link encap:Ethernet  HWaddr 07:80:56:B7:91:8A
     inet addr:10.0.2.43  Bcast:10.0.2.255  Mask:255.255.255.0
```

168页

```
shell# yum install telnet-server
shell# vim /etc/xinetd.d/telnet
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        disable         = no
}
shell# /etc/init.d/xinetd start
Starting xinetd:                                           [  OK  ]
shell# telnet 127.0.0.1 #输入用户名和密码，进行测试
# 如果希望用root登录，则需要配置
```

```
shell# vim /etc/securetty
tty8
tty9
tty10
tty11
pts/0
shell# useradd admin
shell# passwd admin #输入密码mypwd@2018
```

169-170页

```
shell#  vim /etc/zabbix/zabbix_server.conf
### Option: ExternalScripts
#       Full path to location of external scripts.
#       Default depends on compilation options.
#       To see the default path run command "zabbix_server --help".
#
# Mandatory: no
# Default:
# ExternalScripts=${datadir}/zabbix/externalscripts

ExternalScripts=/etc/zabbix/externalscripts

shell# systemctl restart zabbix-server
shell# vim  /etc/zabbix/externalscripts/echo.sh
#!/bin/bash
echo "$1" "$2"
shell# chmod 755 /etc/zabbix/externalscripts/echo.sh
shell# chown zabbix:zabbix /etc/zabbix/externalscripts/echo.sh
```

172页

```
shell# vim /etc/zabbix/zabbix_server.conf
### Option: StartPreprocessors
#       Number of pre-forked instances of preprocessing workers.
#       The preprocessing manager process is automatically started when
preprocessor worker is started.
#
# Mandatory: no
# Range: 1-1000
# Default:
# StartPreprocessors=3
```

177页

https://www.zabbix.com/documentation/4.0/manual/appendix/macros/supported_by_location#additional_support_for_user_macros