keepalived-1.2.3 配置文件关键字官方文档详解翻译中文版

版本 V1.0
时间 2012-07-17
版权 GPL
作者 itnihao 邮箱 itnihao@qq.com
本文档来自 keepalived-1.2.3\doc 官方文档的翻译，如有不妥之处，请邮件联系我，谢谢！
说明:为了更好理解，本文档在原文的基础上给出适当的翻译，请以原文为主，中文适当参考，同时给出了部分原文没有的示例。
本文档不定期更新，地址 http://itnihao.blog.51cto.com，敬请关注

This file describe all the Keepalived available keywords. The keepalived.conf file is compounded by three configurations parts :

此文档描述了所有 keepalived 的关键字，keepalived.conf 文件由三个配置部分组成：

* Globals configurations
* VRRP configuration
* LVS configuration

*全局配置
*VRRP 配置
*LVS 配置

0. Comment string

0.注释字符串

There is 2 valid comment valid string : # or ! If you want to add comment in you configuration file use this char.

这儿有 2 个可用的注释字符串：#或!，在配置文件里面注释的时候，可用这 2 个其中的一个字符串

1. Globals configurations

1.全局配置部分

This block is divided in 2 sub-block :

此块被划分为 2 分块：

* Global definitions
* Static routes

*全局定义部分
*静态路由设置部分

1.1. Global definitions

1.1 全局定义部分

The configuration block looks like :

配置模块如下所示

```
global_defs {                  # Block identification
    #全局配置块
    notification_email {        # Email to send alertes to
    #发生事件时发送的邮件警告
        <EMAIL ADDRESS>     # Standard email address
    #标准的邮件地址，用于接收信息
        <EMAIL ADDRESS>
        ...
    }
    notification_email_from <EMAIL ADDRESS> # Email From dealing with SMTP proto
```
#使用 smtp 协议的邮件

```
    smtp_server <IP ADDRESS>              # SMTP server IP address
```
#SMTP 服务器的地址

```
    smtp_connect_timeout <INTEGER>    # Number of seconds timeout connect remote SMTP server
```
#连接远程 smtp 服务器超时的时间值（单位秒）

```
    router_id <STRING>                    # String identifying router
```
#在路由(vrrp)信息中的字符串（虚拟路由标识 virtual_router_id.这个标识是一个数字，并且同一个 vrrp 实例使用唯一的标识。即同一个 vrrp_stance,MASTER 和 BACKUP 的 virtual_router_id 是一致的，同时在整个 vrrp 内是唯一的。）

```
}
```
#此处的配置实例（注意，此处为原文档没有的）

```
global_defs{
notification_email{
itnihao@qq.com
itnihao@admin.com
}
notification_email_from    itnihao@server.com
smtp_server127.0.0.1
smtp_connect_timeout30
router_id    LVS_DEVEL
}
```

```
linkbeat_use_polling      # Use media link failure detection polling fashion
```
#链路失效的时候采取的措施

1.2. Static addresses

1.2 静态地址

The configuration block looks like :

此模块的配置如下

```
static_ipaddress {                  # block identification
    <IP ADDRESS>/<MASK> brd <IP ADDRESS> dev <STRING> scope <SCOPE>
    <IP ADDRESS>/<MASK> brd <IP ADDRESS> dev <STRING> scope <SCOPE>
    ...
}
```
#此处的配置实例（注意，此处为原文档没有的）

```
static_ipaddress {
      10.10.10.10/24 brd 10.10.10.255 dev eth1 scope global
      20.20.20.20/24 brd 20.20.20.255 dev eth2 scope global
}
```

SCOPE can take the following values :

scope 可以使用如下的选项

* site
* link
* host

* nowhere
* global

1.3. Static routes

1.3 静态路由

The configuration block looks like :

此模块的配置如下

```
static_routes {                          # block identification
    src <IP ADDRESS> [to] <IP ADDRESS>/<MASK> via|gw <IP ADDRESS> dev <STRING> scope <SCOPE> table <TABLE-ID> # to is optional
    src <IP ADDRESS> [to] <IP ADDRESS>/<MASK> via|gw <IP ADDRESS> dev <STRING> scope <SCOPE> table <TABLE-ID> # to is optional
    src <IP ADDRESS> [to] <IP ADDRESS>/<MASK> via|gw <IP ADDRESS> or <IP ADDRESS> dev <STRING> scope <SCOPE> table <TABLE-ID> # will use multipath route
    blackhole <IP ADDRESS>[/<MASK>]
    ...
}
```

#此处的配置实例（注意，此处为原文档没有的）

```
static_routes {
      src   10.10.10.10   to 192.168.1.0/24 via 192.168.1.1 dev eth0 scope
      src   20.20.20.20   to 192.168.2.0/24 gw 192.168.2.1 dev eth1 scope
      src   30.30.30.30   to 192.168.3.0/24 via eth3
      40.40.40.40 dev eth4
      50.50.50.50 via 192.168.5.1/24
}
```

SCOPE can take the following values :

scope 可以使用如下的选项

* site
* link
* host
* nowhere
* global

2. VRRP configuration

2.VRRP 配置（部分）

This block is divided in 3 sub-block :

这一模块分为三个子块

* VRRP scripts
* VRRP synchronization group
* VRRP instance

2.1. VRRP scripts

2.1VRRP 脚本

The configuration block looks like :

此模块的配置如下

```
vrrp_script <STRING> {                  # VRRP script declaration
                                        #VRRP 脚本的定义值
    script <QUOTED_STRING>              # script to run periodically
                                        #脚本的存放路径
    interval <INTEGER>                  # run the script this every seconds
                                        #每隔多少秒运行此脚本
    weight <INTEGER:-254..254>          # adjust priority by this weight
                                        #权重的优先设置
    fall <INTEGER>                      # required number of failures for OK switch
                                        #健康切换当失效后的次数
    rise <INTEGER>                      # required number of successes for OK switch
                                        #健康切换的重试的连接次数
}
```

The script will be executed periodically, every <interval> seconds. Its exit code will be recorded for all VRRP instances which will want to monitor it. Note that the script will only be executed if at least one VRRP instance monitors it with a non-zero weight. Thus, any number of scripts may be declared without taking the system down.

If unspecified, the weight equals 2, which means that a success will add +2 to the priority of all VRRP instances which monitor it. On the opposite, a negative weight will be subtracted from the initial priority in case of failure.

脚本将定期每<interval>秒执行。它的退出代码将记录所有 VRRP 将要监视的实例。请注意，该脚本在至少有一个备份实例监控与非权重为零的情况下才会被执行，。因此，任何数量的脚本在执行的时候都不会导致系统失效。

如果未指定，重量等于 2，这意味着，被监控的健康机器的优先级增加+2。相反，在失败的情况下，将权重量减去最初优先级。

2.2. VRRP synchronization group

2.2VRRP 同步组

The configuration block looks like :

此模块的配置如下

```
vrrp_sync_group <STRING> {      # VRRP sync group declaration
    group {                 # group of instance to sync together
        <STRING>            #    a
        <STRING>            #         set
        ...           #               of VRRP_Instance string
    }
    notify_master <STRING>|<QUOTED-STRING> # Script to run during MASTER transit
    notify_backup <STRING>|<QUOTED-STRING> # Script to run during BACKUP transit
    notify_fault <STRING>|<QUOTED-STRING>   # Script to run during FAULT transit
    notify <STRING>|<QUOTED-STRING>         # Script to run during ANY state transit (1)
    smtp_alert              # Send email notif during state transit
}
```

#此处的配置实例（注意，此处为原文档没有的）

```
vrrp_syncv_group   VG_1{
group{
inside_network
outside_network
}
notify_master      /path/to/to_master.sh
notify_backup      /path/to/to_backup.sh
notify_fault       "/ path/fault.shVG_1"
notify             /path/to/notify.sh
smtp_alter
}
```

#notify_mater 指定当切换到 Master 时，指定的脚本，这个脚本可以传入参数（引号引起），其他 2 个类推

#notify 指定有 3 个参数，这些参数由 Keepalived 提供：

$1(GROUP-INSTANCE),
$2(group 或者 instance 的名字),
$3(MASTER_BACKUP-FAULT)

#smtp_alter 使用 global_defs 里面定义的邮件地址和 smtp 服务器在切换后发送邮件通知

(1) The "notify" script is called AFTER the corresponding notify_* script has been called, and is given exactly 3 arguments (the whole string is interpreted as a litteral filename so don't add parameters!):

$1 = A string indicating whether it's a "GROUP" or an "INSTANCE"
$2 = The name of said group or instance
$3 = The state it's transitioning to ("MASTER", "BACKUP" or "FAULT")

$1 and $3 are ALWAYS sent in uppercase, and the possible strings sent are the same ones listed above ("GROUP"/"INSTANCE", "MASTER"/"BACKUP"/"FAULT").

Important: for a SYNC group to run reliably, it is vital that all instances in the group are MASTER or that they are all either BACKUP or FAULT. A situation with half instances having higher priority on machine A half others with higher priority on machine B will lead to constant re-elections. For this reason, when instances are grouped, their tracking weights are automatically set to zero, in order to avoid inconsistent priorities across instances.

2.3. VRRP instance

2.3 VRRP 实例

The configuration block looks like :

此模块的配置如下

vrrp_instance <STRING> {   # VRRP instance declaration

#vrrp 实例定义

```
    use_vmac                    # Use VRRP Virtual MAC
    #使用虚拟 mac
    native_ipv6                 # Force instance to use IPv6 when using mixed IPv4&IPv6 conf

      #ip4 和 ip6 混合配置时候强制使用 IPV6
    state MASTER|BACKUP                     # Start-up default state
     #设置角色
    interface <STRING>                      # Binding interface
     #监听的端口
    track_interface {                 # Interfaces state we monitor
    #状态监听端口
      <STRING>
      <STRING>
      <STRING> weight <INTEGER:-254..254>
     #权重设置
      ...
    }
    track_script {                          # Scripts state we monitor
    #状态监听脚本
      <STRING>
      <STRING> weight <INTEGER:-254..254>
     #权重设置
      ...
    }
    dont_track_primary                        # (default unset) ignore VRRP interface faults.
                                                #    useful for cross-connect VRRP config.
    mcast_src_ip <IP ADDRESS>               # src_ip to use into the VRRP packets
    #发送多播包的地址，如果不设置，默认使用绑定的网卡的 primaryIP
    lvs_sync_daemon_interface <STRING>       # Binding interface for lvs syncd
    #lvssyncd 绑定的网卡
    garp_master_delay <INTEGER>  # delay for gratuitous ARP after MASTER state transition
    #在切换到 MASTER 状态后，延迟进行 gratuitousARP 请求
    virtual_router_id <INTEGER-0..255>  # VRRP VRID
    #VRID 标记（0...255）
    priority <INTEGER-0..255>         # VRRP PRIO
    #优先级 0-255，数值越大优先级越高
    advert_int <INTEGER>        # VRRP Advert interval (use default)
    #检查间隔，默认 1s
    authentication {                  # Authentication block
    #认证
         auth_type PASS|AH      # Simple Passwd or IPSEC AH
#认证的方式，支持 PASS 和 AH
         auth_pass <STRING>            # Password string
#认证的密码
```

```
    }
    virtual_ipaddress {              # VRRP IP addres block
    #指定漂移地址
        <IP ADDRESS>/<MASK> brd <IP ADDRESS> dev <STRING> scope <SCOPE> label
<LABEL>
        <IP ADDRESS>/<MASK> brd <IP ADDRESS> dev <STRING> scope <SCOPE> label
<LABEL>
        ...
    }
    virtual_ipaddress_excluded {              # VRRP IP excluded from VRRP
        <IP ADDRESS>/<MASK> brd <IP ADDRESS> dev <STRING> scope <SCOPE>   #
packets
        <IP ADDRESS>/<MASK> brd <IP ADDRESS> dev <STRING> scope <SCOPE>
        ...
    }
    virtual_routes {                 # VRRP virtual routes
     #发生切换的时候添加/删除路由
        src <IP ADDRESS> [to] <IP ADDRESS>/<MASK> via|gw <IP ADDRESS> dev
<STRING> scope <SCOPE> table <TABLE-ID> metric <METRIC> # to is optional
        src <IP ADDRESS> [to] <IP ADDRESS>/<MASK> via|gw <IP ADDRESS> dev
<STRING> scope <SCOPE> table <TABLE-ID> metric <METRIC> # to is optional
        src <IP ADDRESS> [to] <IP ADDRESS>/<MASK> via|gw <IP ADDRESS> or <IP
ADDRESS> dev <STRING> scope <SCOPE> table <TABLE-ID> # will use multipath route
        ...
    blackhole <IP ADDRESS>[/<MASK>]
    }
    nopreempt                          # Override VRRP RFC preemption default
    # 不抢占，只在优先级高的机器上设置即可，优先级低的机器不设置
    preempt_delay                    # Seconds after startup until preemption. 0 (default) to 1,000
    # 抢占延迟，默认为 0
    debug                            # Debug level #调试级别
    notify_master <STRING>|<QUOTED-STRING># Same as vrrp_sync_group
    notify_backup <STRING>|<QUOTED-STRING>      # Same as vrrp_sync_group
    notify_fault <STRING>|<QUOTED-STRING>   # Same as vrrp_sync_group
    notify_stop <STRING>|<QUOTED-STRING>    # Script to launch when stopping vrrp
    notify <STRING>|<QUOTED-STRING>         # Same as vrrp_sync_group
    smtp_alert                         # Same as vrrp_sync_group
}
```

SCOPE can take the following values :

* site
* link
* host
* nowhere

* global

LABEL is optional and creates a name for the alias. For compatibility with "ifconfig", it should be of the form <realdev>:<anytext>, for example eth0:1 for an alias on eth0.

LABEL 选项可以建立一个别名，例如用“ifconfig”命令，将会显示<realdev>:<anytext>这种格式，例如 eth0:1 是 eth0 的别名

METRIC is optional and specify a route priority.

METRIC 这个选项可以指定路由的优先级

When a weight is specified in track_interface, instead of setting the vrrp instance to the FAULT state in case of failure, its priority will be increased by the weight when the interface is up (for positive weights), or decreased by the weight's absolute value when the interface is down (for negative weights). The weight must be comprised between -254 and +254 inclusive. 0 is the default behaviour which means that a failure implies a FAULT state. The common practise is to use positive weights to count a limited number of good services so that the server with the highest count becomes master. Negative weights are better to count unexpected failures among a high number of interfaces, as it will not saturate even with high number of interfaces.

The same principle can be applied to track_script entries, except that an unspecified weight means that the default weight declared in the script will be used.

当在 track_interface 里面指定了权重，而不是在失效的时候在 vrrp instance 里面使用 FAULT。接口 up 的时候，权重的优先级将增加（正权重），在接口 down 的时候，权重的绝对值将下降（负权重）。权重的值必须在-254 和+254 范围之间。0 是默认的值，这意味着在失效的时候进入ＦＡＵＬＴ状态。常见的做法是为一个工作正常的服务设置一个正向权重，拥有最高权重的服务将成为 master，负权重在高权重接口发生意外故障时启用，它的值不会设置为比高权重数接口的权重值大。

3. LVS configuration

#LVS 的配置

This block is divided in 2 sub-block :

此模块分为 2 个子模块：

* Virtual server group

* Virtual server

*虚拟服务器组

*虚拟服务器

3.1. Virtual server group

3.1 虚拟服务组

The configuration block looks like :

此模块的配置如下

```
virtual_server_group <STRING> {
        <IP ADDRESS> <PORT>             # VIP VPORT
        <IP ADDRESS> <PORT>
        ...
        <IP ADDRESS RANGE> <PORT>       # VIP range VPORT
        <IP ADDRESS RANGE> <PORT>
```

```
    ...
    fwmark <INTEGER>          # fwmark
    fwmark <INTEGER>
    ...
}
```

Note: <IP ADDRESS RANGE> has the form of : XXX.YYY.ZZZ.WWW-VVV, define the IP address range starting at WWW and monotonaly incremented by one to VVV. Example : 192.168.200.1-10 means .1 to .10 IP addresses.

3.2. Virtual server

3.2 虚拟服务器

The configuration block looks like :

此模块的配置如下

A virtual_server can be either :

一个 virtual_server 可以做如下设置：

* vip vport declaration
* fwmark declaration
* group declaration

```
virtual_server <IP ADDRESS> <PORT> {   # VS IP/PORT declaration
#虚拟服务的 ip/port
virtual_server fwmark <INTEGER>     {   # VS fwmark declaration
virtual_server group <STRING>       {   # VS group declaration
    delay_loop <INTEGER>            # delay timer for service polling
#每隔多少秒检查一次后端的服务器
    lvs_sched rr|wrr|lc|wlc|lblc|sh|dh  # LVS scheduler used
#LVS 的调度算法
    lvs_method NAT|DR|TUN           # LVS method used
#LVS 的工作模式
    persistence_timeout <INTEGER>   # LVS persistence timeout
#LVS 的会话保持时间
    persistence_granularity <NETMASK>  # LVS granularity mask
    protocol TCP                # Only TCP is implemented
#使用协议 TCP
    ha_suspend                 # If VS IP address is not set, suspend healthcheckers activity
    virtualhost <STRING>       # VirtualHost string to use for HTTP_GET or SSL_GET
# virtualhost 字符串用于做 HTTP_GET 或 SSL_GET 检查
```

# Assume silently all RSs down and healthchecks failed on start. This helps preventing false positive actions on startup. Alpha mode is disabled by default.

启动的时候检查后端的服务器或健康检查失败。这有助于防止启动时的失败的情况。alpha 模式默认情况下是禁用的。

alpha

# On daemon shutdown, consider quorum and RS down notifiers for execution, where appropriate. Omega mode is disabled by default.

在守护进程关闭，执行裁决，在真实服务器 down 的时候执行操作并发送通知。omega 模式默认情况下禁用。

omega

# Minimum total weight of all live servers in the pool necessary to operate VS with noquality regression. Defaults to 1.

所有在池中存活的服务器必要的经营与 noquality 回归 VS 服务器的最低总重量。默认为 1

quorum <INT>

# Tolerate this much weight units compared to the nominal quorum, when considering quorum gain or loss. A flap dampener. Defaults to 0.

hysteresis <INT>

# Script to launch when quorum is gained.

quorum_up <STRING>|<QUOTED-STRING>

# Script to launch when quorum is lost.

quorum_down <STRING>|<QUOTED-STRING>

sorry_server <IP ADDRESS> <PORT> # RS to add to LVS topology when all realserver are down

#当 LVS 模型中的真实服务器全部 down 掉的时候将被使用

```
    real_server <IP ADDRESS> <PORT> {       # RS declaration
        weight <INTEGER>             # weight to use (default: 1)
```

#权重（默认是 1）

```
        inhibit_on_failure          # Set weight to 0 on healthchecker    failure
```

#在服务器健康检查失败时，将 weight 设置为 0，而不是直接从 IPVS 里面删除

notify_up <STRING>|<QUOTED-STRING> #Script to launch when healthchecker consider service as up.

#检测到服务器 up 后执行的脚本

notify_down <STRING>|<QUOTED-STRING> #Script to launch when healthchecker consider service as down.

#检测到服务 down 后执行的脚本

```
        HTTP_GET|SSL_GET {            # HTTP and SSL healthcheckers
```

#HTTP/SSL 检查的 URL

```
            url {                     # A set of url to test
              path <STRING>           # Path
```

#路径

```
              digest <STRING>         # Digest computed with genhash
```

#SSL 检查后的校验值（genhash 工具算出）

```
              status_code <INTEGER>   # status code returned into the HTTP header.
```

#HTTP 检查的返回状态码

```
            }
          url {
              path <STRING>
              digest <STRING>
```

```
                    status_code <INTEGER>
                  }
                  ...
                  connect_port <PORT>          # TCP port to connect
 #监控检查端口
                  bindto <IP ADDRESS>          # IP address to bind to
#以此地址发送请求对服务器进行监控检查
                  connect_timeout <INTEGER>       # Timeout connection
#连接超时时间
                  nb_get_retry <INTEGER>      # number of get retry
#重试次数
                  delay_before_retry <INTEGER> # delay before retry
#重连间隔时间
              }
          }

          real_server <IP ADDRESS> <PORT> {       # Idem
              weight <INTEGER>             # Idem
#权重，默认为 1；0 为失效
              inhibit_on_failure          # Idem
#在服务器健康检查失败时，将 weight 设置为 0
              notify_up <STRING>|<QUOTED-STRING> # Idem
#检测到服务器 up 后执行的脚本
              notify_down <STRING>|<QUOTED-STRING> # Idem
#检测到服务器 down 后执行的脚本

              TCP_CHECK {                  # TCP healthchecker
                  connect_port <PORT>          # TCP port to connect
#监控检查端口
                  bindto <IP ADDRESS>          # IP address to bind to
#以此地址发送请求对服务器进行监控检查
                  connect_timeout <INTEGER>       # Timeout connection
#连接超时时间
              }
          }

          real_server <IP ADDRESS> <PORT> {       # Idem
              weight <INTEGER>             # Idem

              inhibit_on_failure          # Idem
              notify_up <STRING>|<QUOTED-STRING> # Idem
              notify_down <STRING>|<QUOTED-STRING> # Idem

              SMTP_CHECK {                          # SMTP healthchecker
```

```
                host {                                  # Optional additional host/port to check
                    connect_ip <IP ADDRESS> # IP address to connect to
                    connect_port <PORT>        # Port to connect to
                    bindto <IP ADDRESS>         # IP address to bind to

                }
                host {
                    connect_ip <IP ADDRESS>
                    connect_port <PORT>
                    bindto <IP ADDRESS>
                }
                ...
                connect_timeout <INTEGER>    # Connection and read/write timeout
                retry <INTEGER>                        # Number of times to retry a failed check
                delay_before_retry <INTEGER> # Delay in seconds before retrying
                helo_name <STRING>|<QUOTED-STRING> # Host to use for the HELO request
            }
        }

        real_server <IP ADDRESS> <PORT> {       # Idem
            weight <INTEGER>              # Idem
            inhibit_on_failure          # Idem
            notify_up <STRING>|<QUOTED-STRING> # Idem
            notify_down <STRING>|<QUOTED-STRING> # Idem

            MISC_CHECK {                        # MISC healthchecker
                misc_path <STRING>|<QUOTED-STRING>#External system script or program
                misc_timeout <INTEGER>          # Script execution timeout
                # If set, exit code from healthchecker is used to dynamically adjust the weight as
follows:
                #     exit status 0: svc check success, weight     unchanged.
                #     exit status 1: svc check failed.
                #     exit status 2-255: svc check success, weight changed to 2 less than exit status.
(for example: exit status of 255 would set weight to 253)
```

如果设置了 misc_dynamic 的话，healthecechker 程序的退出状态码会用来动态调整服务器的权重
返回 0：健康检查 OK，权重不被修改
返回 1：健康检查失败
返回 2-255，健康检查 OK，权重设置为：退出状态码-2，比如返回 255，那么 weight=255-2=253

```
                misc_dynamic
            }
        }
}
```