

Boundary Conditions

by Roland Hedberg (Umeå University), Torbjörn Wiberg (Umeå University)

Table of contents

| | |
|--|---|
| 1 The General Boundary Condition Format..... | 2 |
| 2 The Flat File Boundary Condition..... | 2 |
| 3 The Time Boundary Condition..... | 2 |
| 4 The GDBM Boundary Condition..... | 3 |
| 5 The LDAPset Boundary Condition..... | 4 |
| 6 The Ipnum Boundary Condition..... | 6 |
| 7 The Lastlogin Boundary Condition..... | 6 |
| 8 The Mail Address Boundary Condition..... | 7 |
| 9 The RBL Boundary Condition..... | 7 |
| 10 The Strmatch Boundary Condition..... | 8 |
| 11 The Difftime Boundary Condition..... | 8 |

1. The General Boundary Condition Format

boundary-condition = type ";" *unitsel ";" typespecific

Complete specification in the [rulefile format definition](#).

2. The Flat File Boundary Condition

Building on the definition in 2.5, this boundary condition is described by:

```

type          = "flatfile"
typespecific  = file [":" keyword [":" value *( "," value )]]
file          = utf8string
keyword       = utf8string      ; keywords are not allowed to start with
a\
'#' se below
value        = utf8string

```

Furthermore the format of the flat file has to adhere to the following format

```

line         = (data / comment) CR
comment      = '#' whatever
whatever     = utf8string
data         = keyword ":" *SP value *( *SP "," *SP value ) *SP
SP           = %x20 / %x09      ; SPACE or HTAB

```

Three cases can appear:

Only file is specified

If the file exists and can be opened for reading, the reference will evaluate to *true*. If the file does not exist or if it cannot be opened for reading, the reference will evaluate to *false*.

File and keyword are specified

If the file contains the specified keyword, the reference will evaluate to *true*, if not it will evaluate to *false*.

File, keyword and one or more values are specified

If the file contains the keyword and if every value specified in the reference also appears as values for that keyword in the file then the reference will evaluate to *true*; otherwise, it will evaluate to *false*.

3. The Time Boundary Condition

This boundary condition can be used when a rule shall only be valid for a limited time, or at recurring intervals.

Boundary Conditions

The ABNF of the reference:

```
type          = "time"
typespecific  = [start] [":" [end] [":" days [":" starttime [":"
endtime]]\
]]
start         = full-date          ; as specified in
Intro to S-expressions
end          = date
days        = ["0"]["1"]["2"]["3"]["4"]["5"]["6"] ; Sun = 0, Mon = 1,
\
...
starttime    = partial-time      ; as specified in
Intro to S-expressions
endtime      = partial-time
```

Example:

```
time:2002-08-01_00:00:00;;12345;08:00:00;17:00:00
```

Which means; starting at 00:00:00 on the 1st of August 2002, every Monday, Tuesday, Wednesday, Thursday and Friday between 08:00:00 and 17:00:00 inclusive, this expression will evaluate to *true*.

4. The GDBM Boundary Condition

The workings of the GDBM boundary condition is very similar to flat file, the ABNF is exactly the same except for 'type'.

```
type          = "gdbm"
typespecific  = file [":" keyword [":" value *( "," value )]]
file          = utf8string
keyword       = utf8string        ; keywords are not allowed to start with
a\
'#'
see
below
value        = utf8string
```

The format of the datum in the GDBM file is supposed to be 'value *(", " value)'.

The semantics are also very similar to flat file; three cases can appear:

Only the file is specified

If the gdbm file exists and is can be opened for reading, the reference will evaluate to *true*. If the file does not exist or if it cannot be opened for reading, the reference will evaluate to *false*.

File and keyword are specified

If the gdbm file contains the specified keyword, the reference will evaluate to *true*. If not, it will evaluate to *false*

File, keyword and one or more values are specified

If the gdbm file contains the keyword and if every value specified in the reference also appears as a value for that keyword in the gdbm file, the reference will evaluate to *true*. Otherwise it will evaluate to *false*.

5. The LDAPset Boundary Condition

If you have lots of information about objects in an LDAP directory, this should be a very useful tool for you.

ABNF for the LDAPset boundary condition:

```

type           = "ldapset"
typespecific  = ldapserver *10[";" DN ] ";" vset
ldapserver    = < login from Section 5 of RFC1738 [RFC1738] >
thisDN        = dn
userDN        = dn
dn            = < distinguishedName from Section 3 of RFC2253 [RFC2253] >
vset          = dnvset / valvset
dnvset        = base
               / "(" dnvset ")"
               / "{" dnvset ext attribute}"
               / dnvset SP conj SP dnvset
               / dnvset ext dnattribute-list
               / "<" dn ">"
               / "\" d                               ; the number refers
bac\
k to the DN's in                                     ; the DN list, 0 is
the\
first
  valvset      = ''' string '''
               / "(" valvset ")"
               / dnvset ext attribute-list
               / valvset SP conj SP valvset
  conj         = "&" / "|"
  ext          = "/" / "%" / "$"                       ; base, onelevel
resp.\
subtree search
  a           = %x41-5A / %x61-7A                       ; lower and upper
case\
ASCII
  d           = %x30-39
  k           = a / d / "-" / ";"
  anhstring   = 1*k
  attribute-list = attribute *["," attribute ]
  dnattribute-list = dnattribute *["," dnattribute ]

```

Boundary Conditions

```
attribute = a [ anstring] ; as defined by
[RFC22\
52]
dnattribute = < any attribute name which have attributetype
distinguishedName (1.3.6.1.4.1.1466.115.121.1.12)
like member, owner, roleOccupant, seeAlso,
modifiersName,\
creatorsName,...>
SP = %x20
```

Example:

```
<cn=Group,dc=minorg,dc=se>/member
```

The set of DNs that appear as attribute values of the member attribute in the entry with the DN "cn=Group,dc=minorg,dc=se". This is really a check for whether the attribute member in "cn=Group, dc=minorg,dc=se" has any values.

```
{\0$mail & "sven.svensson@minorg.se"}/title & "mib"
```

The above can be thought of as being evaluated in two steps:

1. Find all the objects in the subtree starting at whatever is given as the first search base DN in the LDAP boundary condition and which has "sven.svensson@minorg.se" as an attribute value for the attribute "mail".
2. Among that set of objects, find the object that has "mib" as an attribute value for the attribute "title".

If this was put into a LDAP filter it would probably be
"(&(mail=sven.svensson@minorg.se)(title=mib))"

```
{\0$mail & "sven.svensson@minorg.se"}/title,personalTitle & "mib"
```

The above can be thought of as being evaluated in two steps:

1. Find all the objects in the subtree starting at whatever is given as the first searchbase DN in the LDAP boundary condition and which has "sven.svensson@minorg.se" as an attribute value for the attribute "mail".
2. Among that set of objects, find the object that has "mib" as an attribute value for either the attributes "title" or "personalTitle".

If this was put into a LDAP filter it would probably be
"(&(mail=sven.svensson@minorg.se)((title=mib)(personalTitle=mib)))"

```
<cn=Group,dc=minorg,dc=se>/member & {\0%mail & "tvw@minorg.se"}
```

Find any object which has the attribute value "tvw@minorg.se" for the "mail" attribute, using a onelevel search below the DN provided as the first searchbase

DN in the LDAP boundary condition. And then check whether any of the DNs of these objects appear as attribute values for the attribute member in the object "cn=Group,dc=minorg,dc=se".

6. The Ipnun Boundary Condition

A special case of the flatfile boundary condition. In this case, the values connected to a key in the flatfile are supposed to be single ipaddresses or groups of ipaddresses. Presently, only ipv4.

The fileformat must be:

```
line      = comment / spec
comment  = '#' whatever
spec     = keyword ':' value *( ',' value )
value    = ipnum "/" netpart
keyword  = utf8string
netpart  = 1*2dig ; min 1 max 32
ipnum    = oct "." oct "." oct "." oct
oct      = 1*3dig ; min 0 max 255
dig      = %x30-39

for instance

umu:130.239.0.0/16,193.193.7.0/24
```

Format of ipnum boundary condition is:

```
type      = "ipnum"
typespecific = file ":" keyword ":" ipnum
file      = utf8string
```

This boundary condition will return TRUE if and only if the file exists, the keyword is present in the file and the ip address in the query matches at least one of the net specifications given as values to the keyword.

7. The Lastlogin Boundary Condition

In order to allow users to use mailrelays when travelling or from their homes, some organizations have used the method where if a user first authenticates using POP or IMAP then within a certain timelimit (20-30 minutes) that user can use the mailrelay. This is a simple backend that knows what the logfile for one popular POP server looks like. It can linearly search this file for the last entry of a specific user and match that timestamp against the present to see if it's within the stipulated time window.

Boundary Conditions

The fileformat is:

```
Jul 25 21:53:25 bassett pop3d: Connection, ip=[::ffff:213.67.231.206]
Jul 25 21:53:25 bassett pop3d: LOGIN, user=sour, ip=[::ffff:213.67.231.206]
Jul 25 21:53:27 bassett pop3d: LOGOUT, user=sour,
ip=[::ffff:213.67.231.206\
], top=0, retr=6188
```

And the boundary condition format is:

```
Will return TRUE if it can find a login in the logs that appeared less
than 'since' ago for the
specified user.
```

8. The Mail Address Boundary Condition

Used when you want to match an email address against a number of mail addresses or mail domain specifications.

File format is :

```
comment = '#' whatever CR
line     = addr-spec / xdomain CR
xdomain = ( "." / "@" ) domain
..
addr-spec and domain as defined by RFC 822
```

Boundary condition format:

```
type      = "addrmatch"
typespec  = file ":" addr-spec
```

Will do a trailing substring match of the specifications found in the file against the address specified in the boundary condition. Will return TRUE if anyone is matched.

9. The RBL Boundary Condition

The Realtime Blackhole List system works by having the ipaddress of the blocked machines stored as A records in DNS using a specific syntax. The names stored in DNS is constructed by reverting the order of the parts in the ip address (130.239.16.3 becomes 3.16.239.130), and then adding a top domain part (blackholes.mail-abuse.org). So, if 130.239.16.3 were to be RBL blacklisted there would be an A record in the DNS with the name 3.16.239.130.blackholes.mail-abuse.org.

If you want to blacklist machines from using services, and you are able to add A records to a DNS server, you can use the same mechanism, -but with your defined top domain part.

The format of the boundary condition:

```
type      = "rbl"  
typespec = domain ":" ipnum  
ipnum     = oct "." oct "." oct "." oct  
oct       = 1*3dig ; min 0 max 255  
dig       = %x30-39
```

Will return TRUE if there is an A record in DNS with a name constructed from 'domain' and 'ipnum' as described above.

10. The Strmatch Boundary Condition

Compares two strings The format of the boundary conditions:

```
type      = "strmatch"  
typespec = string:string[:offset[:num]]  
string    = UTF-8 string  
offset    = number  
num       = number  
number    = 1*dig  
dig       = %x30-39
```

11. The Difftime Boundary Condition

Checks the present time against a given time. There are four different conditions that can be checked:

- If the present time is after a specified time and if the difference exceeds a certain value.
- If the present time is after a specified time and if the difference is less than a certain value.
- If the present time is before a specified time and if the difference exceeds a certain value.
- If the present time is before a specified time and if the difference is less than a certain value.

The format of the boundary condition is:

```
difftspec = diff ";" date ";" when how_much  
date      = YYMMDDTHH:MM:SS  
diff      = YYMMDD_HH:MM:SS  
when      = before / after  
before    = "-"  
after     = "+"
```

Boundary Conditions

```
how_much = less / more  
less     = "-"  
more     = "+"
```

For instance,

000007_00:00:00;2003-10-09T08:00:00;--+

is TRUE if now is before 08:00:00 the 2003-10-09 and if now is more then 7 days before 08:00:00 the 2003-10-09.