
dnsbl Sendmail milter - Version 6.23

Packages

The various source and binary packages are available at <http://www.five-ten-sg.com/dnsbl/packages/>. The most recent documentation is available at <http://www.five-ten-sg.com/dnsbl/>.

A Mercurial [<http://www.selenic.com/mercurial/wiki/>] source code repository for this project is available at <http://hg.five-ten-sg.com/dnsbl/>.

Name

dnsbl -- a sendmail milter with per-user dnsbl filtering

dnsbl

Synopsis

dnsbl [-c] [-s] [-d *n*] [-e *from/to*] [-b *local-domain-socket*] [-r *local-domain-socket*]
[-p *sendmail-socket*] [-t *timeout*]

Options

-c

Load the configuration file, print a canonical form of the configuration on stdout, and exit.

-s

Stress test the configuration loading code by repeating the load/free cycle in an infinite loop.

-d *n*

Set the debug level to *n*.

-e *from/to*

Print the results of looking up the from and to addresses in the current configuration. The | character is used to separate the from and to addresses in the argument to the -e switch.

-b *local-domain-socket-file-name*

Set the local socket used for the connection to the dccifd daemon. This is typically /var/dcc/dccifd.

-r *local-domain-socket-file-name*

Set the local socket used for the connection to our own dns resolver processes.

-p *sendmail-socket*

Set the socket used for the milter connection to sendmail. This is either "inet:port@ip-address" or "local:local-domain-socket-file-name".

-t *timeout*

Set the timeout in seconds used for communication with sendmail.

Usage

dnsbl -c

dnsbl -s

dnsbl -e 'someone@aol.com|localname@mydomain.tld'

dnsbl -d 10 -r resolver.sock -p local:dnsbl.sock

Installation

This is now a standard GNU autoconf/automake installation, so the normal `./configure; make; su; make install` works. `"make chkconfig"` will setup the `init.d` runlevel scripts. Alternatively, you can use the source or binary RPMs at <http://www.five-ten-sg.com/dnsbl/packages>.

Note that this has ONLY been tested on Linux, specifically RedHat Linux. In particular, this milter makes no attempt to understand IPv6. Your mileage will vary. You will need at a minimum a C++ compiler with a minimally thread safe STL implementation. The distribution includes a `test.cpp` program. If it fails this milter won't work. If it passes, this milter might work.

Modify your `sendmail.mc` by removing all the `"FEATURE(dnsbl)"` lines, add the following line in your `sendmail.mc` and rebuild the `.cf` file

```
INPUT_MAIL_FILTER(`dnsbl', `S=local:/var/run/dnsbl/dnsbl.sock, F=T,↵  
T=C:30s;S:5m;R:5m;E:5m')
```

Modify the default `dnsbl.conf(5)` configuration.

Configuration

The configuration file is documented in `dnsbl.conf(5)`. Any change to the config file, or any file included from that config file, will cause it to be reloaded within three minutes.

Introduction

Consider the case of a mail server that is acting as secondary MX for a collection of clients, each of which has a collection of mail domains. Each client may use their own collection of DNSBLs on their primary mail server. We present here a mechanism whereby the backup mail server can use the correct set of DNSBLs for each recipient for each message. As a side-effect, it gives us the ability to customize the set of DNSBLs on a per-recipient basis, so that `fred@example.com` could use LOCAL and the SBL, where all other users `@example.com` use only the SBL.

This milter can also verify the envelope from/recipient pairs with the primary MX server. This allows the backup mail servers to properly reject mail sent to invalid addresses. Otherwise, the backup mail servers will accept that mail, and then generate a bounce message when the message is forwarded to the primary server (and rejected there with no such user). These rejections are the primary cause of such backscatter.

This milter will also decode (uuencode, base64, mime, html entity, url encodings) and scan for HTTP and HTTPS URLs and bare hostnames in the body of the mail. If any of those host names have A or NS records on the SBL (or a single configurable DNSBL), the mail will be rejected unless previously whitelisted. This milter also counts the number of invalid HTML tags, and can reject mail if that count exceeds your specified limit.

This milter can also impose hourly rate limits on the number of recipients accepted from SMTP AUTH connections, that would otherwise be allowed to relay thru this mail server with no spam filtering.

Consider the case of a message from A to B passing thru this milter. If that message is not blocked, then we might eventually see a reply message from B to A. If the filtering context for A includes an autowhite entry, and that context does *not* cover B as a recipient, then this milter will add an entry in that file to whitelist such replies for a configurable time period. Suppose A and B are in the same domain, or at least use the same filtering context. In that case we don't want to add a whitelist entry for B, since that would then allow spammers to send mail from B (forged) to B. Such autowhite files need to be writeable by the dnsbl user, where all the other dnsbl configuration files only need to be readable by the dnsbl user.

You can manually add such an autowhite entry, by appending a single text line to the autowhitelist file, using something like `echo "$mail 0" >>$autowhitefile`. You can manually remove such an autowhite entry, by appending a single text line to the autowhitelist file, using something like `echo "$mail 1" >>$autowhitefile`.

The DNSBL milter reads a text configuration file (dnsbl.conf) on startup, and whenever the config file (or any of the referenced include files) is changed. The entire configuration file is case insensitive. If the configuration cannot be loaded due to a syntax error, the milter will log the error and quit. If the configuration cannot be reloaded after being modified, the milter will log the error and send an email to root from dnsbl@\$hostname. You probably want to add dnsbl@\$hostname to your /etc/mail/virtusertable since otherwise sendmail will reject that message.

DCC Issues

If you are also using the DCC [<http://www.rhyolite.com/anti-spam/dcc/>] milter, there are a few considerations. You may need to whitelist senders from the DCC bulk detector, or from the DNS based lists. Those are two very different reasons for whitelisting. The former is done thru the DCC whiteclnt config file, the later is done thru the DNSBL milter config file.

You may want to blacklist some specific senders or sending domains. This could be done thru either the DCC (on a global basis, or for a specific single recipient). We prefer to do such blacklisting via the DNSBL milter config, since it can be done for a collection of recipient mail domains. The DCC approach has the feature that you can capture the entire message in the DCC log files. The DNSBL milter approach has the feature that the mail is rejected earlier (at RCPT TO time), and the sending machine just gets a generic "550 5.7.1 no such user" message.

The DCC whiteclnt file can be included in the DNSBL milter config by the dcc_to and dcc_from statements. This will import the (env_to, env_from, and substitute mail_host) entries from the DCC config into the DNSBL config. This allows using the DCC config as the single point for white/blacklisting.

Consider the case where you have multiple clients, each with their own mail servers, and each running their own DCC milters. Each client is using the DCC facilities for envelope from/to white/blacklisting. Presumably you can use rsync or scp to fetch copies of your clients DCC whiteclnt files on a regular basis. Your mail server, acting as a backup MX for your clients, can use the DNSBL milter, and include those client DCC config files. The envelope from/to white/blacklisting will be appropriately tagged and used only for the domains controlled by each of those clients.

You can now use (via dccifd) different dcc filtering parameters on a per context basis. See the dcc_greylst and dcc_bulk_threshold statements in the dnsbl.conf(5) configuration. Those statements are only active if you supply the -b option on the dnsbl command line. If you use the dcc via the standard dcc milter (dccm), then connections from clients that use SMTP AUTH are still subject to greylisting. If you use the dcc via dccifd and this milter, then connections from clients that use SMTP AUTH are never subject to greylisting. As part of this per-user greylisting, you need to move the dnsblnogrey file from the config directory to something like /var/dcc/userdirs/local/dnsblnogrey/whiteclnt so the dccifd will properly ignore greylisting for those recipients that don't want it.

Definitions

CONTEXT - a collection of parameters that defines the filtering context to be used for a collection of envelope recipient addresses. The context includes such things as the list of DNSBLs to be used, and the various content filtering parameters.

DNSBL - a named DNS based blocking list is defined by a dns suffix (e.g. sbl-xbl.spamhaus.org) and a message string that is used to generate the "550 5.7.1" smtp error return code. The names of these DNSBLs will be used to define the DNSBL-LISTS.

DNSBL-LIST - a named list of DNSBLs that will be used for specific recipients or recipient domains.

Filtering Procedure

The SMTP envelope 'from' and 'to' values are used in various checks. The first check is to see if a reply message (swapping the env_from and env_to values) would be unconditionally blocked (just based on the envelope from address). That check is similar to the main check described below, but there is no body content to be scanned, and there is no client connection ip address to be checked against DNSBLs. If such a reply message would be blocked, we also block the original outgoing message. This prevents folks from sending mail to recipients that are unable to reply.

If the client has authenticated with sendmail, the rate limits are checked. If the authenticated user has not exceeded the hourly rate limit, then the mail is accepted, the filtering contexts are not used, the dns lists are not checked, and the body content is not scanned. If the client has not authenticated with sendmail, we follow these steps for each recipient.

1. The envelope to email address is used to find an initial filtering context. We first look for a context that specified the full email address in the env_to statement. If that is not found, we look for a context that specified the entire domain name of the envelope recipient in the env_to statement. If that is not found, we look for a context that specified the user@ part of the envelope recipient in the env_to statement. If that is not found, we use the first top level context defined in the config file.
2. The initial filtering context may redirect to a child context based on the values in the initial context's env_from statement. We look for [1) the full envelope from email address, 2) the domain name part of the envelope from address, 3) the user@ part of the envelope from address] in that context's env_from statement, with values that point to a child context. If such an entry is found, we switch to that child filtering context.
3. We lookup [1) the full envelope from email address, 2) the domain name part of the envelope from address, 3) the user@ part of the envelope from address] in the filtering context env_from statement. That results in one of (white, black, unknown, inherit).
4. If the answer is black, mail to this recipient is rejected with "no such user", and the dns lists are not checked.
5. If the answer is white, mail to this recipient is accepted and the dns lists are not checked.
6. If the answer is unknown, we don't reject yet, but the dns lists will be checked, and the content may be scanned.
7. If the answer is inherit, we repeat the envelope from search in the parent context.
8. If the mail has not been accepted or rejected yet, and the filtering context (or any ancestor context) specifies a non-empty whitelist regular expression, then we check the envelope from value against that regex. The mail is accepted if the envelope from value matches the specified regular expression.
9. If the mail has not been accepted or rejected yet, the dns lists specified in the filtering context are checked and the mail is rejected if any list has an A record for the standard dns based lookup scheme (reversed octets of the client followed by the dns suffix).

10. If the mail has not been accepted or rejected yet, and the filtering context (or any ancestor context) specifies a non-empty generic regular expression, then we check the fully qualified client name (obtained via the sendmail macro "_"). The mail is rejected if the client name matches the specified regular expression.
11. If the mail has not been accepted or rejected yet, we look for a verification context, which is the closest ancestor of the filtering context that both specifies a verification host, and which covers the envelope to address. If we find such a verification context, and the verification host is not our own hostname, we open an smtp conversation with that verification host. The current envelope from and recipient to values are passed to that verification host. If we receive a 5xy response those commands, we reject the current recipient with "no such user".
12. If the mail has not been accepted or rejected yet, and the filtering context enables content filtering, and this is the first such recipient in this smtp transaction, we set the content filtering parameters from this context, and enable content filtering for the body of this message.

For each recipient that was accepted, we search for an autowhite entry starting in the reply filtering context. If an autowhite entry is found, we add the recipient to that auto whitelist file. This will prevent reply messages from being blocked by the dnsbl or content filtering.

If content filtering is enabled for this body, the mail text is decoded (uuencode, base64, mime, html entity, url encodings), and scanned for HTTP and HTTPS URLs or bare host names. Hostnames must be either ip address literals, or must end in a string defined by the TLD list. The first <configurable> host names are checked as follows.

The only known list that is suitable for the content filter DNSBL is the SBL. If the content filter DNSBL is defined, and any of those host names resolve to ip addresses that are on that DNSBL (or have nameservers that are on that list), and the host name is not on the <configurable> ignore list, the mail is rejected.

If the content uribl DNSBL is defined, and any of those host names are on that DNSBL, and the host name is not on the <configurable> ignore list, the mail is rejected.

If any non-whitelisted recipient has a filtering context with a non-zero spamassassin limit, then the message is passed thru spamassassin (via spamc), and the message is rejected for those recipients with spamassassin limits less than the resulting spamassassin score. For example, a spamassassin limit of three will reject messages with spamassassin scores of four or greater.

If any non-whitelisted recipient has a filtering context that specifies DCC greylisting, then the message is passed thru the DCC bulk detector, and the message is greylisted (for all recipients) if the DCC says this message should be delayed.

If any non-whitelisted recipient has a filtering context with a non-zero DCC bulk threshold, then the message is passed thru the DCC bulk detector, and the message is rejected for those recipients with DCC thresholds less than or equal to the DCC bulk score.

We also scan for excessive bad html tags, and if a <configurable> limit is exceeded, the mail is rejected.

Sendmail access vs. DNSBL

With the standard sendmail.mc dnsbl FEATURE, the dnsbl checks may be suppressed by entries in the /etc/mail/access database. For example, suppose you control a /18 of address space, and have allocated some /24s to some clients. You have access entries like

```
192.168.4   OK
192.168.17  OK
```

to allow those clients to smarthost thru your mail server. Now if one of those clients happens get infected with a virus that turns a machine into an open proxy, and their 192.168.4.45 lands on the SBL-XBL, you will still wind up allowing that infected machine to smarthost thru your mail servers.

With this DNSBL milter, the sendmail access database cannot override the dnsbl checks, so that machine won't be able to send mail to or thru your smarthost mail server (unless the virus/proxy can use smtp-auth).

Using the standard sendmail features, you would add access entries to allow hosts on your local network to relay thru your mail server. Those OK entries in the sendmail access database will override all the dnsbl checks. With this DNSBL milter, you will need to have the local users authenticate with smtp-auth to get the same effect. You might find these directions [<http://www.ists.dartmouth.edu/classroom/sendmail-ssl-how-to.php>] helpful for setting up smtp-auth if you are on RH Linux.

Performance Issues

Consider a high volume high performance machine running sendmail. Each sendmail process can do its own dns resolution. Typically, such dns resolver libraries are not thread safe, and so must be protected by some sort of mutex in a threaded environment. When we add a milter to sendmail, we now have a collection of sendmail processes, and a collection of milter threads.

We will be doing a lot of dns lookups per mail message, and at least some of those will take many tens of seconds. If all this dns work is serialized inside the milter, we have an upper limit of about 25K mail messages per day. That is clearly not sufficient for many sites.

Since we want to do parallel dns resolution across those milter threads, we add another collection of dns resolver processes. Each sendmail process is talking to a milter thread over a socket, and each milter thread is talking to a dns resolver process over another socket.

Suppose we are processing 20 messages per second, and each message requires 20 seconds of dns work. Then we will have 400 sendmail processes, 400 milter threads, and 400 dns resolver processes. Of course that steady state is very unlikely to happen.

Rejected Ideas

The following ideas have been considered and rejected.

Add max_recipients setting to the context configuration. Recipients in excess of that limit will be rejected, and all the non-whitelisted recipients will be removed. Current spammers *very* rarely send more than ten recipients in a single smtp transaction, so this won't stop any significant amount of spam.

Add poison addresses to the configuration. If any recipient is poison, all recipients are rejected even if they would be whitelisted, and the data is rejected if sent. I have a collection of spam trap addresses that would be suitable for such use. Based on my log files, any mail to those spam trap addresses is rejected based on either dnsbl lookups or the DCC. So this won't result in blocking any additional spam.

Add an option to only allow one recipient if the return path is empty. Based on my log files, there is no mail that violates this check.

Reject the mail if the envelope from domain name contains any MX records pointing to 127.0.0.0/8. I don't see any significant amount of spam sent with such domain names.

TODO

The following ideas are under consideration.

Look for href="hostname/path" strings that are missing the required http:// protocol header. Such references are still clickable in common mail software.

Copyright

Copyright (C) 2007 by 510 Software Group <carl@five-ten-sg.com>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3, or (at your option) any later version.

You should have received a copy of the GNU General Public License along with this program; see the file COPYING. If not, please write to the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.

Version

6.23

Name

dnsbl.conf -- configuration file for dnsbl sendmail milter

dnsbl.conf

Synopsis

dnsbl.conf

Description

The **dnsbl.conf** configuration file is specified by this partial bnf description. Comments start with // or # and extend to the end of the line. To include the contents of some file verbatim in the dnsbl.conf file, use

```
include "<file>";
```

```
CONFIG      = {CONTEXT ";" }+
CONTEXT     = "context" NAME "{" {STATEMENT}+ "}"
STATEMENT   = (DNSBL      | DNSBLLIST | CONTENT | ENV-TO   | VERIFY | GENERIC
               | W_REGEX  | AUTOWHITE | CONTEXT | ENV-FROM | RATE-LIMIT) ";"

DNSBL       = "dnsbl" NAME DNSPREFIX ERROR-MSG1

DNSBLLIST   = "dnsbl_list" {NAME}+

CONTENT     = "content" ("on" | "off") "{" {CONTENT-ST}+ "}"
CONTENT-ST  = (FILTER | URIBL | IGNORE | TLD      | CCTLD  | HTML-TAGS |
               HTML-LIMIT | HOST-LIMIT | SPAMASS | REQUIRE | DCCGREY |
               DCCBULK) ";"

FILTER      = "filter" DNSPREFIX ERROR-MSG2
URIBL       = "uribl"  DNSPREFIX ERROR-MSG3
IGNORE      = "ignore"  "{" {HOSTNAME [";"]}+ "}"
TLD         = "tld"      "{" {TLD      [";"]}+ "}"
CCTLD       = "cctld"    "{" {TLD      [";"]}+ "}"
HTML-TAGS   = "html_tags" "{" {HTMLTAG  [";"]}+ "}"
ERROR-MSG1  = string containing exactly two %s replacement tokens
               both are replaced with the client ip address
ERROR-MSG2  = string containing exactly two %s replacement tokens
               the first is replaced with the hostname, and the second
               is replaced with the ip address
ERROR-MSG3  = string containing exactly two %s replacement tokens
               both are replaced with the hostname

HTML-LIMIT  = "html_limit" ("on" INTEGER ERROR-MSG | "off")

HOST-LIMIT  = "host_limit" ("on" INTEGER ERROR-MSG | "off" |
                           "soft" INTEGER)

SPAMASS     = "spamassassin"      INTEGER
REQUIRE    = "require_match"     ("yes" | "no")
DCCGREY     = "dcc_greylist"      ("yes" | "no")
DCCBULK     = "dcc_bulk_threshold" (INTEGER | "many" | "off")
```

```

ENV-TO      = "env_to"      "{ " {(TO-ADDR | DCC-TO)}+ " }"
TO-ADDR     = ADDRESS [ ";" ]
DCC-TO      = "dcc_to" ("ok" | "many") "{ " DCCINCLUDEFILE " }" ";"

VERIFY      = "verify" HOSTNAME ";"
GENERIC     = "generic" REGULAREXPRESSION ERROR-MSG4 ";"
W-REGEX     = "white_regex" REGULAREXPRESSION ";"
ERROR-MSG4  = string containing exactly one %s replacement token
              which is replaced with the client name
AUTOWHITE   = "autowhite" DAYS FILENAME ";"

ENV_FROM    = "env_from" [DEFAULT] "{ " {(FROM-ADDR | DCC-FROM)}+ " }"
FROM-ADDR   = ADDRESS VALUE [ ";" ]
DCC-FROM    = "dcc_from" "{ " DCCINCLUDEFILE " }" ";"

RATE-LIMIT  = "rate_limit" [DEFAULTLIMIT] "{ " (RATE)+ " }"
RATE        = USER LIMIT [ ";" ]

DEFAULT     = ("white" | "black" | "unknown" | "inherit" | "")
ADDRESS     = (USER@ | DOMAIN | USER@DOMAIN)
VALUE       = ("white" | "black" | "unknown" | "inherit" | CHILD-CONTEXT-NAME)

```

Sample

```

context main-default {
    // outbound dnsbl filtering to catch our own customers that end up on the sbl
    dnsbl sbl sbl-xbl.spamhaus.org "Mail from %s rejected - sbl; see
http://www.spamhaus.org/query/bl?ip=%s";
    dnsbl_list sbl;

    // outbound content filtering to prevent our own customers from sending spam
    content on {
        filter sbl-xbl.spamhaus.org "Mail containing %s rejected - sbl;
see http://www.spamhaus.org/query/bl?ip=%s";
        uribl multi.surbl.org "Mail containing %s rejected - surbl;
see http://www.rulesemporium.com/cgi-bin/uribl.cgi?bl0=1&domain0=%s";
        #uribl black.uribl.com "Mail containing %s rejected - uribl;
see http://l.uribl.com/?d=%s";
        ignore { include "hosts-ignore.conf"; };
        tld { include "tld.conf"; };
        cctld { include "cctld.conf"; };
        html_tags { include "html-tags.conf"; };
        html_limit on 20 "Mail containing excessive bad html tags rejected";
        html_limit off;
        host_limit on 20 "Mail containing excessive host names rejected";
        host_limit soft 20;
        spamassassin 4;
        require_match yes;
        dcc_greylist yes;
        dcc_bulk_threshold 50;
    };
}

```

```

// backscatter prevention - don't send bounces for mail that we accepted but
could not forward
// we only send bounces to our own customers
env_from unknown {
    "<>"    black;
};

// hourly recipient rate limit by smtp auth client id
rate_limit 30 { // default
    #fred 100;    // override default limits
    #joe 10;     // ""
};

context main {
    dnsbl localp partial.blackholes.five-ten-sg.com "Mail from %s rejected -
local; see http://www.five-ten-sg.com/blackhole.php?%s";
    dnsbl local blackholes.five-ten-sg.com "Mail from %s rejected - local;
see http://www.five-ten-sg.com/blackhole.php?%s";
    dnsbl sbl zen.spamhaus.org "Mail from %s rejected - sbl; see
http://www.spamhaus.org/query/bl?ip=%s";
    dnsbl xbl xbl.spamhaus.org "Mail from %s rejected - xbl; see
http://www.spamhaus.org/query/bl?ip=%s";
    dnsbl_list local sbl;

    content on {
        filter sbl-xbl.spamhaus.org "Mail containing %s rejected - sbl;
see http://www.spamhaus.org/query/bl?ip=%s";
        uribl multi.surbl.org "Mail containing %s rejected - surbl;
see http://www.rulesemporium.com/cgi-bin/uribl.cgi?bl0=1&domain0=%s";
        #uribl black.uribl.com "Mail containing %s rejected - uribl;
see http://l.uribl.com/?d=%s";
        ignore { include "hosts-ignore.conf"; };
        tld { include "tld.conf"; };
        cctld { include "cctld.conf"; };
        html_tags { include "html-tags.conf"; };
        html_limit off;
        host_limit soft 20;
        spamassassin 5;
        require_match yes;
        dcc_greylist yes;
        dcc_bulk_threshold 20;
    };

    generic
    "^dsl.static.*ttnet.net.tr$|^([x.-])(ppp|h|host)?([0-9]{1,3}[x.-](Red-|dynamic[x.-])?)$}{4}

    "your mail server %s seems to have a generic name";

    white_regex ".*=example.com=user@yourhostingaccount.com";

    env_to {
        # !! replace this with your domain names

```

```
# child contexts are not allowed to specify recipient addresses outside
these domains
# if this is a backup-mx, you need to include here domains for which you
relay to the primary mx
include "/etc/mail/local-host-names";
};

context whitelist {
    content off {};
    env_to {
        # dcc_to ok { include "/var/dcc/whitecommon"; };
    };
    env_from white {};      # white forces all unmatched from addresses
(everyone in this case) to be whitelisted
                                # so all mail TO these env_to addresses is accepted
};

context abuse {
    dnsbl_list xbl;
    content off {};
    generic "^$" " " " ";    # regex cannot match, to disable generic rdns rejects
    env_to {
        abuse@           # no content filtering on abuse reports
        postmaster@       # "
    };
    env_from unknown {};    # ignore all parent white/black listing
};

context minimal {
    dnsbl_list sbl;
    content on {
        spamassassin      10;
        dcc_bulk_threshold many;
    };
    generic "^$" " " " ";    # regex cannot match, to disable generic rdns rejects
    env_to {
    };
};

context blacklist {
    env_to {
        # dcc_to many { include "/var/dcc/whitecommon"; };
    };
    env_from black {};      # black forces all unmatched from addresses
(everyone in this case) to be blacklisted
                                # so all mail TO these env_to addresses is rejected
};

env_from unknown {
    abuse@ abuse; # replies to abuse reports use the abuse context
    # dcc_from { include "/var/dcc/whitecommon"; };
};

autowhite 90 "autowhite/my-auto-whitelist";
```

```
# install should create /etc/dnsbl/autowhite writable by userid dnsbl  
};
```

Version

6.23