

# GNU LibAntispam 1.2.0 API Reference Manual

Copyright ©2002-2010 Rafael Jorge Csura Szendrodi - Released under LGPL v3  
Manual version 1.5

Generated by Doxygen 1.3.9.1

Sun Nov 8 21:05:15 2009



# Contents

<b>1</b>	<b>GNU LibAntispam 1.2.0 API Data Structure Index</b>	<b>1</b>
1.1	GNU LibAntispam 1.2.0 API Data Structures . . . . .	1
<b>2</b>	<b>GNU LibAntispam 1.2.0 API File Index</b>	<b>3</b>
2.1	GNU LibAntispam 1.2.0 API File List . . . . .	3
<b>3</b>	<b>GNU LibAntispam 1.2.0 API Data Structure Documentation</b>	<b>5</b>
3.1	a_ew_status_s Struct Reference . . . . .	5
3.2	control_user_search_r_s Struct Reference . . . . .	6
3.3	multi_dnsbl_s Struct Reference . . . . .	7
3.4	multi_rsbl_search_s Struct Reference . . . . .	8
3.5	prohibited_ip_net_r_s Struct Reference . . . . .	9
3.6	rsbl_search_s Struct Reference . . . . .	10
3.7	SzAntispam Class Reference . . . . .	11
3.8	SzAntispamMTA Class Reference . . . . .	19
3.9	v_helo_r_s Struct Reference . . . . .	37
3.10	v_mail_from_s Struct Reference . . . . .	38
<b>4</b>	<b>GNU LibAntispam 1.2.0 API File Documentation</b>	<b>39</b>
4.1	antispam++ File Reference . . . . .	39
4.2	antispam++.h File Reference . . . . .	40
4.3	antispam.h File Reference . . . . .	41
4.4	antispam_error.h File Reference . . . . .	49
4.5	mta-heuristics++ File Reference . . . . .	55
4.6	mta-heuristics++.h File Reference . . . . .	56
4.7	mta-heuristics.h File Reference . . . . .	57



# Chapter 1

## GNU LibAntispam 1.2.0 API Data Structure Index

### 1.1 GNU LibAntispam 1.2.0 API Data Structures

Here are the data structures with brief descriptions:

<b>a_ew_status_s</b> (Store the error and warnings codes called LibAntispam function. The structure for LibAntispam error and warnings ) . . . . .	5
<b>control_user_search_r_s</b> (Store the results for a search in control-users list. The structure for antispam heuristics for a search in control-users list ) . . . . .	6
<b>multi_dnsbl_s</b> (Store the results for a multi-dnsbl search. The structure for antispam heuristics mta multi_dnsbl search result ) . . . . .	7
<b>multi_rsbl_search_s</b> (Store the results for multi dnsbl search. The structure for multi dnsbl search functions ) . . . . .	8
<b>prohibited_ip_net_r_s</b> (Store the results for check prohibited IP and Networks. The structure for antispam heuristics check prohibited IP/Networks result ) . .	9
<b>rsbl_search_s</b> (Store the results for a single dnsbl search. The structure for a single dnsbl search functions ) . . . . .	10
<b>SzAntispam</b> (Szendrodi Antispam Class front-end to Szendrodi Antispam Functions )	11
<b>SzAntispamMTA</b> (Antispam MTA Class front-end to Antispam MTA heuristics functions ) . . . . .	19
<b>v_helo_r_s</b> (Store the results for a bad helo ckeck. The structure for antispam heuristics mta helo check result ) . . . . .	37
<b>v_mail_from_s</b> (Store the results for a mail from ckeck. The structure for antispam heuristics mta mail from check result ) . . . . .	38



# Chapter 2

## GNU LibAntispam 1.2.0 API File Index

### 2.1 GNU LibAntispam 1.2.0 API File List

Here is a list of all documented files with brief descriptions:

<b>antispam++</b> (For compatibility with default use of namespace by GCC-3.x series and up ) . . . . .	39
<b>antispam++.h</b> (Universal Szendrodi Antispam API Class methods ) . . . . .	40
<b>antispam.h</b> (Universal Szendrodi Antispam API Routines in C ) . . . . .	41
<b>antispam_error.h</b> (LibAntispam Global Errors and Warning Variables and its Functions ) . . . . .	49
<b>mta-heuristics++</b> (For compatibility with default use of namespace by GCC-3.x series )	55
<b>mta-heuristics++.h</b> (Antispam heuristics API Class (Wild West Antispam Rules) for MTAs ) . . . . .	56
<b>mta-heuristics.h</b> (Antispam heuristics API routines (Wild West Antispam Rules) for MTAs ) . . . . .	57



# Chapter 3

## GNU LibAntispam 1.2.0 API Data Structure Documentation

### 3.1 a\_ew\_status\_s Struct Reference

Store the error and warnings codes called LibAntispam function. The structure for LibAntispam error and warnings.

```
#include <antispam_error.h>
```

#### Data Fields

- int `antispam_error`
- int `antispam_warning`

#### 3.1.1 Detailed Description

Store the error and warnings codes called LibAntispam function. The structure for LibAntispam error and warnings.

#### 3.1.2 Field Documentation

##### 3.1.2.1 int a\_ew\_status\_s::antispam\_error

Antispam error integer.

##### 3.1.2.2 int a\_ew\_status\_s::antispam\_warning

Antispam warning integer.

The documentation for this struct was generated from the following file:

- `antispam_error.h`

## 3.2 control\_user\_search\_r\_s Struct Reference

Store the results for a search in control-users list. The structure for antispam heuristics for a search in control-users list.

```
#include <mta-heuristics.h>
```

### Data Fields

- `uint64_t global_user_search_return_code`
- `int global_user_search_position`
- `global_user_search_type_t global_user_search_type`
- `a_boolean_t global_user_found_in_reduced_format`

### 3.2.1 Detailed Description

Store the results for a search in control-users list. The structure for antispam heuristics for a search in control-users list.

See also:

`Sz_MTA_Heuristics_Search_In_Controlled_Users_List()`(p. 80)

### 3.2.2 Field Documentation

#### 3.2.2.1 `a_boolean_t control_user_search_r_s::global_user_found_in_reduced_format`

User found in reduced (login username only) format.

#### 3.2.2.2 `int control_user_search_r_s::global_user_search_position`

The relative position of the user in some lists used by some LibAntispam functions.

#### 3.2.2.3 `uint64_t control_user_search_r_s::global_user_search_return_code`

Global user search return code that is shared with several LibAntispam functions.

#### 3.2.2.4 `global_user_search_type_t control_user_search_r_s::global_user_search_type`

Global user search type that is used with several LibAntispam functions.

The documentation for this struct was generated from the following file:

- `mta-heuristics.h`

## 3.3 multi\_dnsbl\_s Struct Reference

Store the results for a multi-dnsbl search. The structure for antispam heuristics mta multi\_dnsbl search result.

```
#include <mta-heuristics.h>
```

### Data Fields

- `dnsbl_r_t` result
- `char * message`
- `char * mta_log_message`

### 3.3.1 Detailed Description

Store the results for a multi-dnsbl search. The structure for antispam heuristics mta multi\_dnsbl search result.

See also:

`Sz_MTA_Heuristics_BlackList()`(p. 68)

### 3.3.2 Field Documentation

#### 3.3.2.1 `char* multi_dnsbl_s::message`

message from DNSBL IN TXT entry

#### 3.3.2.2 `char* multi_dnsbl_s::mta_log_message`

message from DNSBL IN TXT entry to write in MTA logs

#### 3.3.2.3 `dnsbl_r_t multi_dnsbl_s::result`

Result of search

See also:

`dnsbl_r_t`(p. ??)

The documentation for this struct was generated from the following file:

- `mta-heuristics.h`

## 3.4 multi\_rsbl\_search\_s Struct Reference

Store the results for multi dnsbl search. The structure for multi dnsbl search functions.

```
#include <antispam.h>
```

### Data Fields

- int **result**
- int **dns\_fail**
- char \* **txt\_message\_ptr**
- char \* **ip\_rsbl\_address\_ptr**

### 3.4.1 Detailed Description

Store the results for multi dnsbl search. The structure for multi dnsbl search functions.

See also:

[Sz\\_Check\\_FQDN\\_or\\_IP\\_in\\_Multiples\\_RSBL\(\)\(p. 43\)](#), [Sz\\_Check\\_FQDN\\_or\\_IPv6\\_in\\_Multiples\\_RSBL\(\)\(p. 45\)](#)

### 3.4.2 Field Documentation

#### 3.4.2.1 int multi\_rsbl\_search\_s::dns\_fail

If temporary DNS fail occurred.

#### 3.4.2.2 char\* multi\_rsbl\_search\_s::ip\_rsbl\_address\_ptr

The IP result of a positive DNSBL query

#### 3.4.2.3 int multi\_rsbl\_search\_s::result

Success or errors, except temporary DNS fail.

#### 3.4.2.4 char\* multi\_rsbl\_search\_s::txt\_message\_ptr

The TXT message for the entry, if result is successful

The documentation for this struct was generated from the following file:

- **antispam.h**

## 3.5 prohibited\_ip\_net\_r\_s Struct Reference

Store the results for check prohibited IP and Networks. The structure for antispam heuristics check prohibited IP/Networks result.

```
#include <mta-heuristics.h>
```

### Data Fields

- `a_boolean_t has_or_belongs_prohibited_ip_or_network`
- `char * deny_ip_network_ptr`

#### 3.5.1 Detailed Description

Store the results for check prohibited IP and Networks. The structure for antispam heuristics check prohibited IP/Networks result.

See also:

`Sz_MTA_Heuristics_Check_For_Prohibited_IP_Network()`(p. 70)

#### 3.5.2 Field Documentation

##### 3.5.2.1 `char* prohibited_ip_net_r_s::deny_ip_network_ptr`

Pointer to denied ip or Network.

##### 3.5.2.2 `a_boolean_t prohibited_ip_net_r_s::has_or_belongs_prohibited_ip_or_network`

Result of the check.

The documentation for this struct was generated from the following file:

- `mta-heuristics.h`

## 3.6 rsbl\_search\_s Struct Reference

Store the results for a single dnsbl search. The structure for a single dnsbl search functions.

```
#include <antispam.h>
```

### Data Fields

- int **result**
- char \* **txt\_message\_ptr**
- char \* **ip\_rsbl\_address\_ptr**

### 3.6.1 Detailed Description

Store the results for a single dnsbl search. The structure for a single dnsbl search functions.

See also:

[Sz\\_Check\\_FQDN\\_or\\_IP\\_in\\_RSBL\(\)](#)(p. 44), [Sz\\_Check\\_FQDN\\_or\\_IPv6\\_in\\_RSBL\(\)](#)(p. 46)

### 3.6.2 Field Documentation

#### 3.6.2.1 char\* rsbl\_search\_s::ip\_rsbl\_address\_ptr

The IP result of a positive DNSBL query

#### 3.6.2.2 int rsbl\_search\_s::result

Success or errors.

#### 3.6.2.3 char\* rsbl\_search\_s::txt\_message\_ptr

The TXT message for the entry, if result is successful

The documentation for this struct was generated from the following file:

- **antispam.h**

## 3.7 SzAntispam Class Reference

Szendrodi Antispam Class front-end to Szendrodi Antispam Functions.

```
#include <antispam++.h>
```

### Public Member Functions

- **int Check\_Valid\_IP\_Number** (char \*ip)  
*Check if an IPv4 number belongs to a valid network (non-RFC1918 and others).*
- **int Check\_Valid\_IPv6\_Number** (char \*ipv6)  
*Check if an IPv6 number belongs to a valid Global unicast network (non Unspecified, Loopback, Multicast, Link-local unicast, Site-local unicast).*
- **int Check\_Valid\_FQDN** (char \*fqdn, a\_boolean\_t ipv6)  
*Check if a FQDN points to a valid IP number (IPv4 or IPv6).*
- **int Check\_Valid\_FQDN\_or\_IP** (char \*fqdn\_or\_ip, a\_boolean\_t ipv6)  
*Check if a FQDN or an IP number (IPv4 or IPv6) is valid.*
- **rsbl\_search\_t Check\_FQDN\_or\_IP\_in\_RSBL** (char \*fqdn\_or\_ip, char \*rsbl=NULL)  
*Check if a FQDN or IPv4 is indexed in a Real-Time IPv4 DNS List.*
- **rsbl\_search\_t Check\_FQDN\_or\_IPv6\_in\_RSBL** (char \*fqdn\_or\_ipv6, char \*rsbl=NULL)  
*Check if a FQDN or IPv6 is indexed in a Real-Time IPv6 DNS List.*
- **multi\_rsbl\_search\_t Check\_FQDN\_or\_IP\_in\_Multiples\_RSBL** (char \*fqdn\_or\_ip, char \*rsbl[])  
*Check if a FQDN or IPv4 is indexed in multiples Real-Time DNS List.*
- **multi\_rsbl\_search\_t Check\_FQDN\_or\_IPv6\_in\_Multiples\_RSBL** (char \*fqdn\_or\_ip, char \*rsbl[])  
*Check if a FQDN or IPv6 is indexed in multiples Real-Time DNS List.*
- **int a\_antispam\_error** (void)  
*Return the value of a LibAntispam internal error number.*
- **int a\_antispam\_warning** (void)  
*Return the value of a LibAntispam internal warning number.*
- **char \* a\_str\_antispam\_error** (void)  
*Return a pointer to LibAntispam internal error message.*
- **char \* a\_str\_antispam\_warning** (void)  
*Return a pointer to LibAntispam internal warning message.*

### 3.7.1 Detailed Description

Szendrodi Antispam Class front-end to Szendrodi Antispam Functions.

### 3.7.2 Member Function Documentation

#### 3.7.2.1 `int SzAntispam::a_antispan_error (void)`

Return the value of a LibAntispam internal error number.

**Returns:**

The value of error number.

#### 3.7.2.2 `int SzAntispam::a_antispan_warning (void)`

Return the value of a LibAntispam internal warning number.

**Returns:**

The value of warning number.

#### 3.7.2.3 `char* SzAntispam::a_str_antispan_error (void)`

Return a pointer to LibAntispam internal error message.

**Returns:**

The error message associated with error number or unknown error message if error number is out of range.

#### 3.7.2.4 `char* SzAntispam::a_str_antispan_warning (void)`

Return a pointer to LibAntispam internal warning message.

**Returns:**

The warning message associated with warning number or unknown warning message if warning number is out of range.

#### 3.7.2.5 `multi_rsbl_search_t SzAntispam::Check_FQDN_or_IP_in_Multiples_RSBL (char * fqdn_or_ip, char * rsbl[])`

Check if a FQDN or IPv4 is indexed in multiples Real-Time DNS List.

**Parameters:**

*fqdn\_or\_ip* pointer to FQDN or IPv4 of the host to verify.

*rsbl* pointer to a null terminated array list of DNSBLs to be consulted.

**Note:**

The pointer array `rsbl` must be terminated by a NULL pointer!

**Returns:**

A `multi_rsbl_search_t` structure with 2 fields.

**Note:**

Field result: Success or errors, except temporary DNS fail.

Position number `N = Host` is indexed in DNS List `N` (where `N` is the ordinal position of the DNS list in the array `rsbl[]`, e.g. 1 if the list is the first DNS list, 2 if is is the second DNS list, ...). The TXT record message is returned in `*txt_message_ptr` and the IP record in DNSBL is returned in `*ip_rsbl_address_ptr`

`A_UNSUCCESS` = Host is not indexed in none DNS list.

**Note:**

Field `dns_fail`: If temporary DNS fail occurred.

Position number `N = DNS fail` in DNS List `N` (where `N` is the ordinal position of the DNS list in the array `rsbl[]`, e.g. 1 if the list is the first DNS list, 2 if is is the second DNS list, ...).

`A_UNSUCCESS` = No DNS fail in none DNS list positions.

**Note:**

If errors occurred, `antispam_error` is set with last error code:

`A_CFIMR_NULL_IP_FQDN` = Null FQDN or IP number passed.

`A_CFIMR_MEMORY_ALLOC_FAIL` = Memory alloc fail.

`A_CFIMR_NO_HOSTENT_ASSOCIATED` = No hostent associated to FQDN.

`A_CFIMR_IP_NUMBER_TOO_SHORT` = IP number too short.

`A_CFIMR_NULL_RSBL_LIST` = Null RSBL list passed.

**Note:**

The array `*rsbl[]` should be terminated by a NULL pointer!

### 3.7.2.6 `rsbl_search_t SzAntispam::Check_FQDN_or_IP_in_RSBL (char * fqdn_or_ip, char * rsbl = NULL)`

Check if a FQDN or IPv4 is indexed in a Real-Time IPv4 DNS List.

**Parameters:**

`fqdn_or_ip` pointer to FQDN or IPv4 of the host to verify.

*rsbl* pointer to DNSBL to be consulted.

**Note:**

If *rsbl* is NULL, default to *rsbl.aupads.org*.

**Returns:**

A\_SUCCESS = Host is indexed The TXT record message is returned in *\*txt\_message\_ptr* and the IP record in DNSBL is returned in *\*ip\_rsbl\_address\_ptr*.

A\_UNSUCCESS = Host is not indexed.

**Note:**

If errors occurred, *antispam\_error* is set with last error code:

A\_CFIR\_NULL\_IP\_FQDN = Null IP number passed.

A\_CFIR\_MEMORY\_ALLOC\_FAIL = Memory alloc fail.

A\_CFIR\_NO\_HOSTENT\_ASSOCIATED = No hostent associated.

A\_CFIR\_IP\_NUMBER\_TOO\_SHORT = IP number too short.

A\_CFIR\_DNS\_TEMP\_FAIL = Temporary DNS fail.

### 3.7.2.7 multi\_rsbl\_search\_t SzAntispam::Check\_FQDN\_or\_IPv6\_in\_Multiples\_RSBL (char \* *fqdn\_or\_ip*, char \* *rsbl[]*)

Check if a FQDN or IPv6 is indexed in multiples Real-Time DNS List.

**Parameters:**

*fqdn\_or\_ip* pointer to FQDN or IPv6 of the host to verify.

*rsbl* pointer to a null terminated array list of DNSBLs to be consulted.

**Note:**

The pointer array *rsbl* must be terminated by a NULL pointer!

**Returns:**

A *multi\_rsbl\_search\_t* structure with 2 fields.

**Note:**

Field result: Success or errors, except temporary DNS fail.

Position number N = Host is indexed in DNS List N (where N is the ordinal position of the DNS list in the array *rsbl[]*, e.g. 1 if the list is the first DNS list, 2 if is the second DNS list, ...). The TXT record message is returned in *\*txt\_message\_ptr* and the IP record in DNSBL is returned in *\*ip\_rsbl\_address\_ptr*

A\_UNSUCCESS = Host is not indexed in none DNS list.

**Note:**

Field `dns_fail`: If temporary DNS fail occurred.

Position number `N` = DNS fail in DNS List `N` (where `N` is the ordinal position of the DNS list in the array `rsbl[]`, e.g. 1 if the list is the first DNS list, 2 if is is the second DNS list, ...).

`A_UNSUCCESS` = No DNS fail in none DNS list.

**Note:**

If errors occurred, `antispam_error` is set with last error code:

`A_CFI6MR_NULL_IP_FQDN` = Null FQDN or IP number passed (on error).

`A_CFI6MR_MEMORY_ALLOC_FAIL` = Memory alloc fail (on error).

`A_CFI6MR_NO_HOSTENT_ASSOCIATED` = No hostent associated to FQDN (on error).

`A_CFI6MR_IP_NUMBER_TOO_SHORT` = IP number too short (on error).

`A_CFI6MR_NULL_RSBL_LIST` = Null RSBL list passed (on error).

**Note:**

The array `*rsbl[]` should be terminated by a NULL pointer!

### 3.7.2.8 `rsbl_search_t SzAntispam::Check_FQDN_or_IPv6_in_RSBL (char * fqdn_or_ipv6, char * rsbl = NULL)`

Check if a FQDN or IPv6 is indexed in a Real-Time IPv6 DNS List.

**Parameters:**

*fqdn\_or\_ipv6* pointer to FQDN or IPv6 of the host to verify.

*rsbl* pointer to DNSBL to be consulted.

**Note:**

If `rsbl` is NULL, default to `ip6.rsbl.aupads.org`.

**Returns:**

`A_SUCCESS` = Host is indexed The TXT record message is returned in `*txt_message_ptr` and the IP record in DNSBL is returned in `*ip_rsbl_address_ptr`.

`A_UNSUCCESS` = Host is not indexed.

**Note:**

If errors occurred, `antispam_error` is set with last error code:

`A_CFI6R_NULL_IP_FQDN` = Null IP number passed.

`A_CFI6R_MEMORY_ALLOC_FAIL` = Memory alloc fail.

A\_CFI6R\_NO\_HOSTENT\_ASSOCIATED = No hostent associated.

A\_CFI6R\_IP\_NUMBER\_TOO\_SHORT = IP number too short.

A\_CFI6R\_DNS\_TEMP\_FAIL = Temporary DNS fail.

A\_CFI6R\_CREATE\_DNSBL\_QUERY\_FAIL = Create IPv6 DNSBL query fail.

### 3.7.2.9 int SzAntispam::Check\_Valid\_FQDN (char \* *fqdn*, a\_boolean\_t *ipv6*)

Check if a FQDN points to a valid IP number (IPv4 or IPv6).

#### Parameters:

*fqdn* A string pointer to FQDN.

*ipv6* A boolean to define if IP is IPv4 (A\_FALSE) or IPv6 (A\_TRUE).

#### Returns:

A\_SUCCESS = Points to a valid IP number.

A\_UNSUCCESS = Points to an invalid IP number.

#### Note:

If errors occurred, `antispam_error` is set with last error code:

A\_CVF\_NULL\_IP\_FQDN = Null FQDN passed.

A\_CVF\_ZERO\_LEN = Zero length FQDN.

A\_CFV\_MEMORY\_ALLOC\_FAIL = Memory alloc fail.

A\_CFV\_NO\_HOSTENT\_ASSOCIATED = No hostent associated.

A\_CVF\_IPV4\_INVALID\_FORMAT = Invalid IPv4 format.

A\_CVF\_IPV6\_INVALID\_FORMAT = Invalid IPv6 format.

### 3.7.2.10 int SzAntispam::Check\_Valid\_FQDN\_or\_IP (char \* *fqdn\_or\_ip*, a\_boolean\_t *ipv6*)

Check if a FQDN or an IP number (IPv4 or IPv6) is valid.

#### Parameters:

*fqdn\_or\_ip* A string pointer to FQDN or IP number.

*ipv6* A boolean to define if IP to be find is IPv4 (A\_FALSE) or IPv6 (A\_TRUE) if string point to an FQDN. Make no sense if string point to an IPv4 or IPv6 number.

**Returns:**

A\_SUCCESS = Valid IP or FQDN points to a valid IP number.

A\_UNSUCCESS = Invalid IP or FQDN points to an invalid IP number.

**Note:**

If errors occurred, antisпам\_error is set with last error code:

A\_CVFI\_NULL\_IP\_FQDN = Null FQDN or IP number passed.

A\_CVFI\_ZERO\_LEN = Zero length FQDN.

A\_CVFI\_MEMORY\_ALLOC\_FAIL = Memory alloc fail.

A\_CVFI\_NO\_HOSTENT\_ASSOCIATED = No hostent associated.

A\_CVFI\_IPV4\_INVALID\_FORMAT = Invalid IPv4 format.

A\_CVFI\_IPV6\_INVALID\_FORMAT = Invalid IPv6 format.

**3.7.2.11 int SzAntispam::Check\_Valid\_IP\_Number (char \* ip)**

Check if an IPv4 number belongs to a valid network (non-RFC1918 and others).

**Parameters:**

*ip* A strings pointer to IPv4 number.

**Returns:**

A\_SUCCESS = Valid IPv4 number.

A\_UNSUCCESS = Invalid IPv4 number.

**Note:**

If errors occurred, antisпам\_error is set with last error code:

A\_CVIN\_OUT\_OF\_RANGE = Octetes out of range.

A\_CVIN\_NULL\_IP\_FQDN = Null IP number passed.

A\_CVIN\_MEMORY\_ALLOC\_FAIL = Memory alloc fail.

A\_CVIN\_NOT\_DOT\_QUAD\_IP = Not a dotted quad IP number associated.

**3.7.2.12** `int SzAntispam::Check_Valid_IPv6_Number (char * ipv6)`

Check if an IPv6 number belongs to a valid Global unicast network (non Unspecified, Loopback, Multicast, Link-local unicast, Site-local unicast).

**Parameters:**

*ipv6* A strings pointer to IPv6 number.

**Returns:**

A\_SUCCESS = Valid IPv6 number (Global unicast)

A\_UNSUCCESS = Invalid IPv6 number (others) or errors (and `antisпам_error` is set)

**Note:**

If errors occurred, `antisпам_error` is set with last error code:

A\_CVIPV6N\_OOMM = Out of memory.

A\_CVIPV6N\_NULL\_IP = Null IPv6 number passed.

A\_CVIPV6N\_INVALID\_FORMAT = Invalid IPv6 format.

**Note:**

See: RFC-3513 for more details

The documentation for this class was generated from the following file:

- `antisпам++.h`

## 3.8 SzAntispamMTA Class Reference

Antispam MTA Class front-end to Antispam MTA heuristics functions.

```
#include <mta-heuristics++.h>
```

### Public Member Functions

- **SzAntispamMTA** (void)
- **bool Heuristics\_Init** (void)  
*Init configs, rejected words and accepted hosts/users lists.*
- **void Heuristics\_Clear** (void)  
*Dealloc memory for rejected words and accepted hosts/users lists and for others init configs.*
- **bool Heuristics\_Rebuild** (void)  
*Rebuild system configurations or control\_user list and private users configurations.*
- **dns\_error\_t Heuristics\_Check\_Reverse\_DNS** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, **control\_user\_search\_r\_t** user\_opt)  
*Check Reverse DNS in agreement with RFC-1912.*
- **char \*Heuristics\_Search\_In\_Rejected\_List** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, **control\_user\_search\_r\_t** user\_opt)  
*Find a domain's substring in rejected words list.*
- **bool Heuristics\_Search\_In\_Accepted\_Hosts\_List** (char \*RFC5321\_Remote\_Hostname\_IP\_Information)  
*Find host in accepted hosts lists.*
- **control\_user\_search\_r\_t Heuristics\_Search\_In\_Controlled\_Users\_List** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Rcpt\_User)  
*Find user in controlled users lists. Return values below are combined and should be verified using AND (&) operation.*
- **bool Heuristics\_Check\_For\_My\_or\_Friend\_IP\_Network** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, **control\_user\_search\_r\_t** user\_opt)  
*Check if remote client is local or friend IP/Network.*
- **v\_mail\_from\_t Heuristics\_Check\_Mail\_From** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, char \*Helo, **control\_user\_search\_r\_t** \*user\_opt)  
*Check for legality of sender user declared in MAIL FROM command.*
- **a\_boolean\_t Heuristics\_Check\_Good\_Mail\_From** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, **control\_user\_search\_r\_t** \*user\_opt)  
*Check if sender user declared in MAIL FROM cmd is a good user.*
- **graylist\_return\_t Heuristics\_GrayList** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, char \*Rcpt\_User, **control\_user\_search\_r\_t** user\_opt)

*Use Advanced GrayList support.*

- **bool Heuristics\_GrayList\_Clean** (void)  
*Clean up the GrayList cache directory and its entries.*
- **multi\_dnsbl\_t Heuristics\_BlackList** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, **control\_user\_search\_r\_t** user\_opt)  
*Use Advanced BlackList support.*
- **int Heuristics\_Check\_Local\_User\_Status\_In\_Mail\_From** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*User\_Address)  
*Verify if user address in MAIL FROM cmd is local and if is Invalid, Restricted or Non-Sender.*
- **int Heuristics\_Check\_Local\_User\_Status\_In\_Rcpt\_To** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*User\_Address)  
*Verify if user address in RCPT TO cmd is local and if is Invalid, Restricted or Non-Sender.*
- **prohibited\_ip\_net\_r\_t Heuristics\_Check\_For\_Prohibited\_IP\_Network** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, **control\_user\_search\_r\_t** user\_opt)  
*Check if remote client is/belongs to a prohibited IP/Network.*
- **bool Heuristics\_Sendmail** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, char \*Rcpt\_User, char \*message, int message\_type, **control\_user\_search\_r\_t** user\_opt)  
*Send a warning mail to somebody.*
- **v\_helo\_r\_t Heuristics\_Helo\_Check** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Helo, **control\_user\_search\_r\_t** user\_opt)  
*Verify HELO SMTP command for bad things.*
- **remote\_smtp\_sender\_return\_t Heuristics\_Check\_Remote\_SMTP\_Sender** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, **control\_user\_search\_r\_t** user\_opt)  
*Verify if domain/MX accept SMTP connection for a sender address.*
- **a\_boolean\_t Heuristics\_Vrfy\_For\_Sender\_User\_Deny\_SMTP\_DATA** (char \*Mail\_From)  
*Verify if the user part name of sender information match with an user, in internal list, that is prohibited to do the DATA SMTP command (should be implemented in MTA, but is quit simple, we need only return a 550 that is the permanent error code in SMTP messages when DATA command is used).*
- **a\_boolean\_t Heuristics\_Vrfy\_For\_Recipient\_User\_Deny\_SMTP\_DATA** (char \*Rcpt\_To)  
*Verify if the user part name of recipient information match with a user, in internal list, that is prohibited to do the DATA SMTP command (should be implemented in MTA, but is quit simple, we need only return a 451 that is the temporary error code in SMTP messages).*
- **int a\_mh\_antispam\_error** (void)  
*Return the value of a LibAntispam internal error number.*

- `int a_mh_antispam_warning` (void)  
*Return the value of a LibAntispam internal warning number.*
- `char * a_mh_str_antispam_error` (void)  
*Return a pointer to LibAntispam internal error message.*
- `char * a_mh_str_antispam_warning` (void)  
*Return a pointer to LibAntispam internal warning message.*
- `~SzAntispamMTA` (void)

### 3.8.1 Detailed Description

Antispam MTA Class front-end to Antispam MTA heuristics functions.

### 3.8.2 Constructor & Destructor Documentation

#### 3.8.2.1 SzAntispamMTA::SzAntispamMTA (void)

Constructor

#### 3.8.2.2 SzAntispamMTA::~~SzAntispamMTA (void)

Destructor

### 3.8.3 Member Function Documentation

#### 3.8.3.1 int SzAntispamMTA::a\_mh\_antispam\_error (void)

Return the value of a LibAntispam internal error number.

**Returns:**

The value of error number.

#### 3.8.3.2 int SzAntispamMTA::a\_mh\_antispam\_warning (void)

Return the value of a LibAntispam internal warning number.

**Returns:**

The value of warning number.

#### 3.8.3.3 char\* SzAntispamMTA::a\_mh\_str\_antispam\_error (void)

Return a pointer to LibAntispam internal error message.

**Returns:**

The error message associated with error number or unknown error message if error number is out of range.

**3.8.3.4 char\* SzAntispamMTA::a\_mh\_str\_antispam\_warning (void)**

Return a pointer to LibAntispam internal warning message.

**Returns:**

The warning message associated with warning number or unknown warning message if warning number is out of range.

**3.8.3.5 multi\_dnsbl\_t SzAntispamMTA::Heuristics\_BlackList (char \* RFC5321\_Remote\_Hostname\_IP\_Information, control\_user\_search\_r\_t user\_opt)**

Use Advanced BlackList support.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

**Returns:**

DNSBL\_IN if IP was found in one BlackList return message in (multi\_dnsbl\_t).message ptr.

DNSBL\_NOT if IP wasn't found in BlackList or on errors.

DNSBL\_TEMP\_FAIL if one of ours dnsbl return message in (multi\_dnsbl\_t).message ptr.

DNSBL\_ERROR on errors and antispam\_error is set.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information MUST has one of the formats below:

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This function is controlled by user options!

**See also:**

**Heuristics\_Search\_In\_Controlled\_Users\_List()**(p. 33)

### 3.8.3.6 `bool SzAntispamMTA::Heuristics_Check_For_My_or_Friend_IP_Network` (char \* *RFC5321\_Remote\_Hostname\_IP\_Information*, control\_user\_search\_r\_t *user\_opt*)

Check if remote client is local or friend IP/Network.

#### Parameters:

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

#### Returns:

A\_TRUE if remote client is local or friend IP/Network.

A\_FALSE if not or on errors.

#### Note:

RFC5321\_Remote\_Hostname\_IP\_Information MUST has one of the formats below:

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

#### Note:

This function is controlled by user options!

#### See also:

`Heuristics_Search_In_Controlled_Users_List()`(p. 33)

### 3.8.3.7 `prohibited_ip_net_r_t SzAntispamMTA::Heuristics_Check_For_Prohibited_IP_Network` (char \* *RFC5321\_Remote\_Hostname\_IP\_Information*, control\_user\_search\_r\_t *user\_opt*)

Check if remote client is/belongs to a prohibited IP/Network.

#### Parameters:

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

**Returns:**

A\_TRUE if remote client is/belongs to a prohibited IP/Network.

In this case, denied IP/Network is pointed by `*deny_ip_network_ptr` pointer.

**Returns:**

A\_FALSE if not or on errors.

**Note:**

if mta heuristics is disabled, return A\_FALSE always.

RFC5321\_Remote\_Hostname\_IP\_Information MUST has one of the formats below:

**IPv4:** [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This function is controled by user options!

**See also:**

`Heuristics_Search_In_Controlled_Users_List()`(p. 33)

**3.8.3.8** `a_boolean_t SzAntispamMTA::Heuristics_Check_Good_Mail_From`  
 (`char * RFC5321_Remote_Hostname_IP_Information`, `char * Mail_From`, `control_user_search_r_t * user_opt`)

Check if sender user declared in MAIL FROM cmd is a good user.

**Parameters:**

`RFC5321_Remote_Hostname_IP_Information` A char string pointer to RFC5321\_Remote\_Hostname\_IP\_Information.

`Mail_From` A char string pointer to MAIL FROM user information.

`user_opt` A `control_user_search_r_t` pointer to rcpt user options.

**Returns:**

A\_TRUE if sender user is a good user.  
 A\_FALSE if not or on errors (and antisipam\_error is set!)

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

Mail\_From SHOULD has the format: user@anything.foo.faa or  
 <user@anything.foo.faa>

### 3.8.3.9 int SzAntispamMTA::Heuristics\_Check\_Local\_User\_Status\_In\_Mail\_From (char \* RFC5321\_Remote\_Hostname\_IP\_Information, char \* User\_Address)

Verify if user address in MAIL FROM cmd is local and if is Invalid, Restricted or Non-Sender.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to RFC5321\_Remote\_Hostname\_IP\_Information.

*User\_Address* A char pointer to sender user address.

**Returns:**

MTA\_USER\_LOCAL\_OK if local user is VALID and not RESTRICT.  
 MTA\_USER\_LOCAL\_INVALID if local user is INVALID.  
 MTA\_USER\_LOCAL\_RESTRICT if local user is RESTRICT.  
 MTA\_USER\_LOCAL\_NON\_SENDER if local user is NON-SENDER user.

**Note:**

This function check internaly the user\_opt.global\_user\_search\_return\_code variable and user\_opt.global\_user\_found\_in\_reduced\_format for the given mail address, if it belongs to our control.users list.

This fuction is a front-end to internal function Sz\_MTA\_Heuristics\_Check\_User\_Status().

### 3.8.3.10 `int SzAntispamMTA::Heuristics_Check_Local_User_Status_In_Rcpt_To` (char \* *RFC5321\_Remote\_Hostname\_IP\_Information*, char \* *User\_Address*)

Verify if user address in RCPT TO cmd is local and if is Invalid, Restricted or Non-Sender.

#### Parameters:

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to RFC5321\_Remote\_Hostname\_IP\_Information.

*User\_Address* A char pointer to recipient user address.

#### Returns:

MTA\_USER\_LOCAL\_OK if local user is VALID and not RESTRICT.

MTA\_USER\_LOCAL\_INVALID if local user is INVALID.

MTA\_USER\_LOCAL\_RESTRICT if local user is RESTRICT.

MTA\_USER\_LOCAL\_NON\_SENDER if local user is NON-SENDER user.

#### Note:

This function check internally the `user_opt.global_user_search_return_code` variable and `user_opt.global_user_found_in_reduced_format` for the given mail address, if it belongs to our control.users list.

This fuction is a front-end to internal function `Sz_MTA_Heuristics_Check_User_Status()`.

### 3.8.3.11 `v_mail_from_t SzAntispamMTA::Heuristics_Check_Mail_From` (char \* *RFC5321\_Remote\_Hostname\_IP\_Information*, char \* *Mail\_From*, char \* *Helo*, `control_user_search_r_t` \* *user\_opt*)

Check for legality of sender user declared in MAIL FROM command.

#### Parameters:

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*Mail\_From* A char string pointer to MAIL FROM user information.

*Helo* A char string pointer to HELO information.

*user\_opt* A `control_user_search_r_t` pointer to rcpt user options.

#### Returns:

V\_MAIL\_FROM\_OK if no problem found in sender information.

V\_MAIL\_FROM\_NULL\_REVERSE\_PATH\_NOT\_LOCAL if null reverse-path "<>" was declared by sender user, but remote client isn't a local or friend machine.

V\_MAIL\_FROM\_LOCAL\_BUT\_CLIENT\_NOT\_LOCAL if sender is local and remote client isn't a local or friend machine.

V\_MAIL\_FROM\_HAS\_PROHIBITED\_CHARS if sender has prohibited characters and `prohibited_char` variable is set.

V\_MAIL\_FROM\_ERROR on errors.

#### Note:

*RFC5321\_Remote\_Hostname\_IP\_Information* MUST has one of the formats below:

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

Mail\_From MUST has the format: `user@anything.foo.faa`  
This function is controlled by user options!

**See also:**

`Heuristics_Search_In_Controlled_Users_List()`(p. 33)

**3.8.3.12 remote\_smtp\_sender\_return\_t SzAntispam-MTA::Heuristics\_Check\_Remote\_SMTP\_Sender (char \* RFC5321\_Remote\_Hostname\_IP\_Information, char \* Mail\_From, control\_user\_search\_r\_t user\_opt)**

Verify if domain/MX accept SMTP connection for a sender address.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to RFC5321\_Remote\_Hostname\_IP\_Information.

*Mail\_From* A char string pointer to address passed in MAIL FROM command.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

**Returns:**

REMOTE\_SMTP\_SENDER\_OK if domain/MX accept sender address.

REMOTE\_SMTP\_SENDER\_FAIL if any fase of SMTP transaction with domain/MX return a fail condition.

REMOTE\_SMTP\_SENDER\_SKIP if this test was skipped due it was disabled by user/admin.

REMOTE\_SMTP\_SENDER\_SENDER\_AUTH if remote host passed in previous authentication check like SPF.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_ADMIN\_REJECT\_IP\_IN\_ADDRESS if remote host used literal IP in MAIL FROM address but the system administrator reject the use of literal IPs in mail addresses.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_NOT\_IN\_LITERAL\_FORMAT if remote host didn't use IP numbers in literal format in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_FORMAT\_INVALID if remote host used IP numbers with invalid format in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_INVALID\_IPV4 if remote host used an invalid IPv4 number in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_INVALID\_IPV4\_RFC1918 if remote host used and private RFC1918 IP number in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_INVALID\_IPV6 if remote host used an invalid IPv6 number in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_IPV6\_ADDR\_IN\_IPV4\_CONNECTION if remote host used an IPv6 number in MAIL FROM address, but the current connection type is IPv4.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_IPV4\_ADDR\_IN\_IPV6\_CONNECTION if remote host used an IPv4 number in MAIL FROM address, but the current connection type is IPv6.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_NOT\_RESOLVE\_TO\_FQDN if remote host used an IP number, in MAIL FROM address, that don't resolve to a FQDN.

REMOTE\_SMTP\_SENDER\_ERROR on errors and a `_err_war->antispam_error` is set!

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**3.8.3.13 dns\_error\_t SzAntispamMTA::Heuristics\_Check\_Reverse\_DNS**  
(char \* *RFC5321\_Remote\_Hostname\_IP\_Information*,  
control\_user\_search\_r\_t *user\_opt*)

Check Reverse DNS in agreement with RFC-1912.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

**Returns:**

DNS\_CHECK\_OK if remote client is inaccordance with RFC-1912.

DNS\_CHECK\_FAIL if remote client IP hasn't a reverse DNS pointer or if reverse DNS information is null.

DNS\_CHECK\_TEMP\_FAIL on DNS temporary fails or on internals errors.

DNS\_CHECK\_FORGED if remote client has a reverse DNS pointer but it don't point to the IP number of the remote client or if it don't point to anything.

DNS\_CHECK\_DISABLED if Reverse DNS check is disabled.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information MUST has one of the formats below:

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This method is controled by user options!

**See also:**

[Heuristics\\_Search\\_In\\_Controlled\\_Users\\_List\(\)](#)(p. 33)

**3.8.3.14 void SzAntispamMTA::Heuristics\_Clear (void)**

Dealloc memory for rejected words and accepted hosts/users lists and for others init configs.

**Returns:**

None.

**3.8.3.15 graylist\_return\_t SzAntispamMTA::Heuristics\_GrayList (char \* RFC5321\_Remote\_Hostname\_IP\_Information, char \* Mail\_From, char \* Rcpt\_User, control\_user\_search\_r\_t user\_opt)**

Use Advanced GrayList support.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*Mail\_From* A char string pointer to MAIL FROM user information.

*Rcpt\_User* A char string pointer to RCPT TO user information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

**Returns:**

GRAYLIST\_IN if IP/SENDER/RCPT was added to GrayList or found and delay time wasn't reached.

GRAYLIST\_OUT if IP/SENDER/RCPT was found and removed from GrayList because delay time was reached if GrayList support is disabled, return A\_FALSE always on errors.

GRAYLIST\_SENDER\_AUTH if SENDER was authenticated previously by any authentication method (like SPF).

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information MUST has one of the formats below:

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

Mail\_From MUST has the format: user@anything.foo.faa.

Rcpt\_User MUST be anything like: anybody@somewhere.foo.net.

This function is controlled by user options!

**See also:**

Heuristics\_Search\_In\_Controlled\_Users\_List()(p. 33)

**3.8.3.16 bool SzAntispamMTA::Heuristics\_GrayList\_Clean (void)**

Clean up the GrayList cache directory and its entries.

**Returns:**

A\_TRUE if clean up with success.

A\_FALSE if not clean up performed.

**3.8.3.17 v\_helo\_r\_t SzAntispamMTA::Heuristics\_Helo\_Check (char \* RFC5321\_Remote\_Hostname\_IP\_Information, char \* Helo, control\_user\_search\_r\_t user\_opt)**

Verify HELO SMTP command for bad things.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*Helo* A char string pointer to HELO information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

**Returns:**

HELO\_OK if helo argument is verified with success.

HELO\_ERROR on errors and a\_err\_war->antispam\_error is set!

HELO\_IP\_NOT\_LITERAL if IP numbers passed is not in literal format.

HELO\_IP\_FORMAT\_INVALID if IP numbers passed has invalid format.  
 HELO\_IPV4\_RFC1918 if IP IPv4 IP number is invalid.  
 HELO\_DOMAIN\_HAS\_PROHIBITED\_CHARS if FQDN has prohibited characters, in case of a prohibited character, the \*prohibited\_char pointer is set.  
 HELO\_DOMAIN\_HAS\_NOT\_DOT if FQDN has not dot points.  
 HELO\_DOMAIN\_START\_WITH\_DOT if FQDN has dot point at the begin.  
 HELO\_DOMAIN\_END\_WITH\_DOT if FQDN has dot point at the end.  
 HELO\_DOMAIN\_RESOLVE\_INVALID\_IPV4 if FQDN points to an invalid IPv4 number.  
 HELO\_DOMAIN\_RESOLVE\_INVALID\_IPV6 if FQDN points to an invalid IPv6 number.  
 HELO\_DOMAIN\_NOT\_RESOLVE\_AND\_NOT\_HAS\_MX if FQDN don't resolve and MX record don't resolve.  
 HELO\_DOMAIN\_RESOLVE\_MX\_INVALID\_IPV4 if MX resolve to an invalid IPv4 number.  
 HELO\_DOMAIN\_RESOLVE\_MX\_INVALID\_IPV6 if MX resolve to an invalid IPv6 number.  
 HELO\_HAS\_PROHIBITED\_IP\_NETWORK if literal IP number match with bad helo list, in this case the denied IP/Network is pointed by \*bad\_helo\_domain\_ptr pointer.  
 HELO\_PROHIBITED\_ALL\_IPV4\_NUMBERS if all literal IPv4 numbers are prohibited in helo command.  
 HELO\_PROHIBITED\_ALL\_IPV6\_NUMBERS if all literal IPv6 numbers are prohibited in helo command.  
 HELO\_HAS\_PROHIBITED\_DOMAIN if FQDN match with bad helo list, in this case of the denied domain is pointed by \*bad\_helo\_domain\_ptr pointer.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

Helo SHOULD be not null string.

This function is not in agreement with RFC-1123!!!

**See also:**

Sz\_MTA\_Heuristics\_Search\_In\_Controlled\_Users\_List()(p.80)

**3.8.3.18 bool SzAntispamMTA::Heuristics\_Init (void)**

Init configs, rejected words and accepted hosts/users lists.

**Returns:**

A\_TRUE if all tasks were performed with success.  
A\_FALSE if get errors, like out of memory.

**3.8.3.19 bool SzAntispamMTA::Heuristics\_Rebuild (void)**

Rebuild system configurations or control\_user list and private users configurations.

**Returns:**

A\_TRUE if all tasks were performed with success.  
A\_FALSE if get errors, like out of memory.

**3.8.3.20 bool SzAntispamMTA::Heuristics\_Search\_In\_Accepted\_Hosts\_List (char \* RFC5321\_Remote\_Hostname\_IP\_Information)**

Find host in accepted hosts lists.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

**Returns:**

A\_TRUE if found a host in accepted hosts list.  
A\_FALSE if not, on errors or if reverse DNS information is null.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information MUST has one of the formats below:

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This method is controled by user options!

**See also:**

[Heuristics\\_Search\\_In\\_Controlled\\_Users\\_List\(\)](#)(p. 33)

### 3.8.3.21 `control_user_search_r_t SzAntispamMTA::Heuristics_Search_In - Controlled_Users_List (char * RFC5321_Remote_Hostname_IP - Information, char * Rcpt_User)`

Find user in controled users lists. Return values below are combinated and should be verified using AND (&) operation.

#### Parameters:

***RFC5321\_Remote\_Hostname\_IP\_Information*** A char string pointer to RFC5321 - Remote\_Hostname\_IP\_Information.

***Rcpt\_User*** A char string pointer to user information.

#### Returns:

**V\_NO\_USER** if no found user in controled users list.

**V\_NONE** if found user and no check will be procced.

**V\_INVALID** if found user and he is an invalid user.

**V\_RESTRICT** if found user and he is a restricted user.

**V\_RESTRICT\_POST** if found user and he has restricted post rights (not implemented yet).

**V\_WARNING\_INSTEAD\_REJECT** if found user and he don't want to reject messages, but add a warning in the subject of message (not implemented yet).

**V\_REVERSE** if found user and check reverse.

**V\_PROHIBITED\_FQDN\_STRINGS** if found user and he want to check if remote host has a prohibited FQDN string in reverse DNS (typically dialup strings), if reverse DNS exist.

**V\_BLACKLIST\_NO** if found user and NO use blacklist.

**V\_BLACKLIST\_YES** if found user and USE blacklist.

**V\_BLACKLIST\_PERSIST** if found user and USE blacklist in PERSIST mode.

**V\_BLACKLIST\_ADMIN\_DECIDES** if found user and USE blacklist in ADMIN\_DECIDES mode.

**V\_GRAYLIST** if found user and use graylist.

**V\_WHITELIST\_SPF\_NO** if found user and he don't want to use whitelist (based on SPF - Sender Policy Framework) (not implemented yet, only for LibAntispam 1.2.0 or up).

**V\_WHITELIST\_SPF\_YES** if found user and he want to use whitelist (not implemented yet, only for LibAntispam 1.2.0 or up).

**V\_WHITELIST\_SPF\_PERSIST** if found user and he want to USE whitelist in PERSISTENT mode (refuse mail while whitelist return TRY\_AGAIN (temporary DNS error)) (not implemented yet, only for LibAntispam 1.2.0 or up).

**V\_WHITELIST\_SPF\_ADMIN\_DECIDES** if found user and he want to USE whitelist in ADMIN\_DECIDES mode (lets system administrator choose if use whitelist with/without PERSISTENT mode or not use whitelist) (not implemented yet, only for LibAntispam 1.2.0 or up).

**V\_WHITELIST\_SPF\_AT\_USER\_LEVEL** if found user and he want to use whitelist SPF at user level (not implemented yet).

**V\_BAD\_HELO** if found user and he want to restrict bad HELO command use.

**V\_REMOTE\_DOMAIN\_SMTP\_SENDER\_CHECKING** if found user and he want to verify if remote domain or its MX accept SMTP connections for sender passed in MAIL FROM cmd. (not implemented yet, only for a future version of LibAntispam).

**V\_FROM** if found user and he want to verify mail from consistency.

**V\_USER\_RESTRICT\_NULL\_REVERSE\_PATH\_USE** if user want to restrict null-reverse path

**V\_USER\_RESTRICT\_LOCAL\_DOMAIN\_USE** if user want to restrict local domain use

**V\_USER\_RESTRICT\_PROHIBITED\_CHARS\_USE** if user want to restrict prohibited chars

**V\_USER\_RESTRICT\_BAD\_MAIL\_FROM\_USE** if user want to restrict bad MAIL FROM cmd (not implemented yet!)

V\_HEADER\_ANALYZE if found user and use realtime header analyze (not implemented yet!)

V\_ERROR on errors (may be synonym to V\_NO\_USER. If list is empty, antispam\_warning is set).

**Note:**

global\_user\_search\_return\_code is set with return combined values above.

Rcpt\_User SHOULD be an E-mail address, like: anybody@somewhere.foo or <anybody@somewhere.foo>.

**3.8.3.22 char\* SzAntispamMTA::Heuristics\_Search\_In\_Rejected\_List**  
**(char \* RFC5321\_Remote\_Hostname\_IP\_Information,**  
**control\_user\_search\_r\_t user\_opt)**

Find a domain's substring in rejected words list.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

**Returns:**

NULL if no domain's substring match with rejected words or on errors.

NULL\_DNS\_INFORMATION if reverse DNS information is null.

SPECIAL\_COMMAND\_RETURN if domain match in special command search and antispam\_warning can be set with two values: SU\_CNS\_FOUND\_NUMERIC\_SETS if an wanted numeric sets were found or SU\_CNG\_FOUND\_NUMERIC\_GROUP if an wanted numeric group was found.

The prohibited word, if it was found in rejected words list.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information MUST has one of the formats below:

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This method is controled by user options!

**See also:**

**Heuristics\_Search\_In\_Controlled\_Users\_List()**(p. 33)

**3.8.3.23** `bool SzAntispamMTA::Heuristics_Sendmail (char * RFC5321_Remote_Hostname_IP_Information, char * Mail_From, char * Rcpt_User, char * message, int message_type, control_user_search_r_t user_opt)`

Send a warning mail to somebody.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*Mail\_From* A char string pointer to MAIL FROM user information.

*Rcpt\_User* A char string pointer to RCPT TO user information.

*message* A char string pointer to warning string message.

*message\_type* A int number to message type.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

**Returns:**

A\_TRUE if mail message was sent with success.

A\_FALSE if not or on errors.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information MUST has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

Mail\_From MUST has the format: `user@anything.faa` or `<user@anything.faa>`.

Rcpt\_User MUST be anything like: `anybody@somewhere.foo` or `<anybody@somewhere.foo>`.

This function is controled by user options!

**See also:**

`Heuristics_Search_In_Controlled_Users_List()`(p. 33)

### 3.8.3.24 `a_boolean_t SzAntispamMTA::Heuristics_Vrfy_For_Recipient_User_Deny_SMTp_DATA (char * Rcpt_To)`

Verify if the user part name of recipient information match with a user, in internal list, that is prohibited to do the DATA SMTP command (should be implemented in MTA, but is quit simple, we need only return a 451 that is the temporary error code in SMTP messages).

#### Parameters:

*Rcpt\_To* A char string pointer to address passed in RCPT TO command.

#### Returns:

A\_TRUE if match with an user in internal list.  
A\_FALSE if not or on errors.

#### Note:

This is a simple function independent from LibAntispam admin or users options, so we don't need to verify things like race or rebuild conditions.  
This function is a front-end to `Sz_MTA_Heuristics_Vrfy_For_User_Deny_SMTp_DATA()`

### 3.8.3.25 `a_boolean_t SzAntispamMTA::Heuristics_Vrfy_For_Sender_User_Deny_SMTp_DATA (char * Mail_From)`

Verify if the user part name of sender information match with an user, in internal list, that is prohibited to do the DATA SMTP command (should be implemented in MTA, but is quit simple, we need only return a 550 that is the permanent error code in SMTP messages when DATA command is used).

#### Parameters:

*Mail\_From* A char string pointer to address passed in MAIL FROM command.

#### Returns:

A\_TRUE if match with an user in internal list.  
A\_FALSE if not or on errors.

#### Note:

This is a simple function independent from LibAntispam admin or users options, so we don't need to verify things like race or rebuild conditions.  
This function is a front-end to `Sz_MTA_Heuristics_Vrfy_For_User_Deny_SMTp_DATA()`

The documentation for this class was generated from the following file:

- `mta-heuristics++.h`

## 3.9 v\_helo\_r\_s Struct Reference

Store the results for a bad helo ckeck. The structure for antispam heuristics mta helo check result.

```
#include <mta-heuristics.h>
```

### Data Fields

- `helo_error_t return_value`
- `char * prohibited_char`
- `char * bad_helo_domain_ptr`

### 3.9.1 Detailed Description

Store the results for a bad helo ckeck. The structure for antispam heuristics mta helo check result.

See also:

`Sz_MTA_Heuristics_Helo_Check()`(p. 78)

### 3.9.2 Field Documentation

#### 3.9.2.1 `char* v_helo_r_s::bad_helo_domain_ptr`

Pointer to a bad HELO found.

#### 3.9.2.2 `char* v_helo_r_s::prohibited_char`

Pointer to a prohibited character found.

#### 3.9.2.3 `helo_error_t v_helo_r_s::return_value`

HELO ckeck return code.

The documentation for this struct was generated from the following file:

- `mta-heuristics.h`

## 3.10 v\_mail\_from\_s Struct Reference

Store the results for a mail from ccheck. The structure for antispam heuristics mta mail from check result.

```
#include <mta-heuristics.h>
```

### Data Fields

- `v_mail_from_return_t` `return_value`
- `char *` `prohibited_char`
- `char` `unwanted_char`
- `char *` `bad_mail_from_domain`
- `char *` `spf_exp`

### 3.10.1 Detailed Description

Store the results for a mail from ccheck. The structure for antispam heuristics mta mail from check result.

See also:

`Sz_MTA_Heuristics_Check_Mail_From()`(p. 73)

### 3.10.2 Field Documentation

#### 3.10.2.1 `char* v_mail_from_s::bad_mail_from_domain`

Pointer to a bad MAIL FROM found.

#### 3.10.2.2 `char* v_mail_from_s::prohibited_char`

Pointer to a prohibited character found.

#### 3.10.2.3 `v_mail_from_return_t v_mail_from_s::return_value`

MAIL FROM ccheck return code.

#### 3.10.2.4 `char* v_mail_from_s::spf_exp`

Pointer to SPF explanation, if it exists.

#### 3.10.2.5 `char v_mail_from_s::unwanted_char`

An unwanted character found in begin or end of mail address.

The documentation for this struct was generated from the following file:

- `mta-heuristics.h`

## Chapter 4

# GNU LibAntispam 1.2.0 API File Documentation

### 4.1 antispam++ File Reference

For compatibility with default use of namespace by GCC-3.x series and up.

```
#include <antispam++.h>
```

#### 4.1.1 Detailed Description

For compatibility with default use of namespace by GCC-3.x series and up.

## 4.2 antispam++.h File Reference

Universal Szendrodi Antispam API Class methods.

```
#include <antispam.h>
```

### Data Structures

- class **SzAntispam**

*Szendrodi Antispam Class front-end to Szendrodi Antispam Functions.*

### 4.2.1 Detailed Description

Universal Szendrodi Antispam API Class methods.

**Author:**

Rafael Jorge Csura Szendrodi <szendro@pads.ufrj.br>

**Date:**

01/01/2002 GNU Lesser General Public License version 3

## 4.3 antispam.h File Reference

Universal Szendrodi Antispam API Routines in C.

```
#include <antispam_error.h>
```

### Data Structures

- struct **rsbl\_search\_s**  
*Store the results for a single dnsbl search. The structure for a single dnsbl search functions.*
- struct **multi\_rsbl\_search\_s**  
*Store the results for multi dnsbl search. The structure for multi dnsbl search functions.*

### Defines

- #define **A\_UNSUCCESS** 0  
*Unsuccess return code. Return unsuccess on function exit or errors.*
- #define **A\_SUCCESS** 1  
*Success return code. Return success on function exit.*

### Typedefs

- typedef **rsbl\_search\_s** **rsbl\_search\_t**  
*Store the results for a single dnsbl search. The typedef-structure for a single dnsbl search functions.*
- typedef **multi\_rsbl\_search\_s** **multi\_rsbl\_search\_t**  
*Store the results for multi dnsbl search. The typedef-structure for multi dnsbl search functions.*

### Enumerations

- enum **a\_boolean\_t** { **A\_FALSE** = 0, **A\_TRUE** = 1 }  
*Antispam heuristics boolean type definition.*

### Functions

- int **Sz\_Check\_Valid\_IP\_Number** (char \*ip, **a\_ew\_status\_t** \*a\_err\_war)  
*Check if an IPv4 number belongs to a valid network (non-RFC1918 and others).*
- int **Sz\_Check\_Valid\_IPv6\_Number** (char \*ipv6, **a\_ew\_status\_t** \*a\_err\_war)  
*Check if an IPv6 number belongs to a valid Global unicast network (non Unspecified, Loopback, Multicast, Link-local unicast, Site-local unicast).*

- `int Sz_Check_Valid_FQDN (char *fqdn, a_boolean_t ipv6, a_ew_status_t *a_err_war)`  
*Check if a FQDN points to a valid IP number (IPv4 or IPv6).*
- `int Sz_Check_Valid_FQDN_or_IP (char *fqdn_or_ip, a_boolean_t ipv6, a_ew_status_t *a_err_war)`  
*Check if a FQDN or an IP number (IPv4 or IPv6) is valid.*
- `rsbl_search_t Sz_Check_FQDN_or_IP_in_RSBL (char *fqdn_or_ip, char *rsbl, a_ew_status_t *a_err_war)`  
*Check if a FQDN or IPv4 is indexed in a Real-Time IPv4 DNS List.*
- `rsbl_search_t Sz_Check_FQDN_or_IPv6_in_RSBL (char *fqdn_or_ip, char *rsbl, a_ew_status_t *a_err_war)`  
*Check if a FQDN or IPv6 is indexed in a Real-Time IPv6 DNS List.*
- `multi_rsbl_search_t Sz_Check_FQDN_or_IP_in_Multiples_RSBL (char *fqdn_or_ip, char *rsbl[], a_ew_status_t *a_err_war)`  
*Check if a FQDN or IPv4 is indexed in multiples Real-Time DNS List.*
- `multi_rsbl_search_t Sz_Check_FQDN_or_IPv6_in_Multiples_RSBL (char *fqdn_or_ip, char *rsbl[], a_ew_status_t *a_err_war)`  
*Check if a FQDN or IPv6 is indexed in multiples Real-Time DNS List.*

## Variables

- `char * txt_message_ptr`  
*Pointer to DNSBL IN TXT message. Pointer to DNSBL IN TXT message.*
- `char * ip_rsbl_address_ptr`  
*Pointer to DNSBL IN A entry. Pointer to DNSBL IN A entry.*

### 4.3.1 Detailed Description

Universal Szendrodi Antispam API Routines in C.

#### Author:

Rafael Jorge Csura Szendrodi <szendro@pads.ufrj.br>

#### Date:

01/01/2002 GNU Lesser General Public License version 3

### 4.3.2 Typedef Documentation

#### 4.3.2.1 struct multi\_rsbl\_search\_t

Store the results for multi dnsbl search. The typedef-structure for multi dnsbl search functions.

See also:

`Sz_Check_FQDN_or_IP_in_Multiples_RSBL()`(p. 43), `Sz_Check_FQDN_or_IPv6_in_Multiples_RSBL()`(p. 45)

#### 4.3.2.2 struct rsbl\_search\_t

Store the results for a single dnsbl search. The typedef-structure for a single dnsbl search functions.

See also:

`Sz_Check_FQDN_or_IP_in_RSBL()`(p. 44), `Sz_Check_FQDN_or_IPv6_in_RSBL()`(p. 46)

### 4.3.3 Function Documentation

#### 4.3.3.1 multi\_rsbl\_search\_t Sz\_Check\_FQDN\_or\_IP\_in\_Multiples\_RSBL(char \* fqdn\_or\_ip, char \* rsbl[], a\_ew\_status\_t \* a\_err\_war)

Check if a FQDN or IPv4 is indexed in multiples Real-Time DNS List.

**Parameters:**

*fqdn\_or\_ip* pointer to FQDN or IPv4 of the host to verify.

*rsbl* pointer to a null terminated array list of DNSBLs to be consulted.

*a\_err\_war* A pointer to error and warning typedef-structure passed by reference.

**Note:**

The pointer array rsbl must be terminated by a NULL pointer!

**Returns:**

A multi\_rsbl\_search\_t structure with 2 fields.

**Note:**

Field result: Success or errors, except temporary DNS fail.

Position number N = Host is indexed in DNS List N (where N is the ordinal position of the DNS list in the array rsbl[], e.g. 1 if the list is the first DNS list, 2 if is is the second DNS list, ...). The TXT record message is returned in \*txt\_message\_ptr and the IP record in DNSBL is returned in \*ip\_rsbl\_address\_ptr.

A\_UNSUCCESS = Host is not indexed in none DNS list.

**Note:**

Field dns\_fail: If temporary DNS fail occurred.

Position number N = DNS fail in DNS List N (where N is the ordinal position of the DNS list in the array rsbl[], e.g. 1 if the list is the first DNS list, 2 if is is the second DNS list, ...).

A\_UNSUCCESS = No DNS fail in none DNS list positions.

**Note:**

If errors occurred, `antispam_error` is set with last error code:

`A_CFIMR_NULL_IP_FQDN` = Null FQDN or IP number passed.

`A_CFIMR_MEMORY_ALLOC_FAIL` = Memory alloc fail.

`A_CFIMR_NO_HOSTENT_ASSOCIATED` = No hostent associated to FQDN.

`A_CFIMR_IP_NUMBER_TOO_SHORT` = IP number too short.

`A_CFIMR_NULL_RSBL_LIST` = Null RSBL list passed.

**Note:**

The array `*rsbl[]` should be terminated by a NULL pointer!

#### 4.3.3.2 `int Sz_Check_FQDN_or_IP_in_RSBL (char * fqdn_or_ip, char * rsbl, a_ew_status_t * a_err_war)`

Check if a FQDN or IPv4 is indexed in a Real-Time IPv4 DNS List.

**Parameters:**

*fqdn\_or\_ip* pointer to FQDN or IPv4 of the host to verify.

*rsbl* pointer to DNSBL to be consulted.

*a\_err\_war* A pointer to error and warning typedef-structure passed by reference.

**Note:**

If *rsbl* is NULL, default to `rsbl.aupads.org`.

**Returns:**

`A_SUCCESS` = Host is indexed. The TXT record message is returned in `*txt_message_ptr` and the IP record in DNSBL is returned in `*ip_rsbl_address_ptr`.

`A_UNSUCCESS` = Host is not indexed.

**Note:**

If errors occurred, `antispam_error` is set with last error code:

`A_CFIR_NULL_IP_FQDN` = Null IP number passed.

`A_CFIR_MEMORY_ALLOC_FAIL` = Memory alloc fail.

`A_CFIR_NO_HOSTENT_ASSOCIATED` = No hostent associated.

`A_CFIR_IP_NUMBER_TOO_SHORT` = IP number too short.

`A_CFIR_DNS_TEMP_FAIL` = Temporary DNS fail.

### 4.3.3.3 multi\_rsbl\_search\_t Sz\_Check\_FQDN\_or\_IPv6\_in\_Multiples\_RSBL (char \* fqdn\_or\_ip, char \* rsbl[], a\_ew\_status\_t \* a\_err\_war)

Check if a FQDN or IPv6 is indexed in multiples Real-Time DNS List.

**Parameters:**

*fqdn\_or\_ip* pointer to FQDN or IPv6 of the host to verify.

*rsbl* pointer to a null terminated array list of DNSBLs to be consulted.

*a\_err\_war* A pointer to error and warning typedef-structure passed by reference.

**Note:**

The pointer array rsbl must be terminated by a NULL pointer!

**Returns:**

A multi\_rsbl\_search\_t structure with 2 fields.

**Note:**

Field result: Success or errors, except temporary DNS fail.

Position number N = Host is indexed in DNS List N (where N is the ordinal position of the DNS list in the array rsbl[], e.g. 1 if the list is the first DNS list, 2 if is is the second DNS list, ...). The TXT record message is returned in \*txt\_message\_ptr and the IP record in DNSBL is returned in \*ip\_rsbl\_address\_ptr.

A\_UNSUCCESS = Host is not indexed in none DNS list.

**Note:**

Field dns\_fail: If temporary DNS fail occurred.

Position number N = DNS fail in DNS List N (where N is the ordinal position of the DNS list in the array rsbl[], e.g. 1 if the list is the first DNS list, 2 if is is the second DNS list, ...).

A\_UNSUCCESS = No DNS fail in none DNS list.

**Note:**

If errors occurred, antispam\_error is set with last error code:

A\_CFI6MR\_NULL\_IP\_FQDN = Null FQDN or IP number passed (on error).

A\_CFI6MR\_MEMORY\_ALLOC\_FAIL = Memory alloc fail (on error).

A\_CFI6MR\_NO\_HOSTENT\_ASSOCIATED = No hostent associated to FQDN (on error).

A\_CFI6MR\_IP\_NUMBER\_TOO\_SHORT = IP number too short (on error).

A\_CFI6MR\_NULL\_RSBL\_LIST = Null RSBL list passed (on error).

**Note:**

The array \*rsbl[] should be terminated by a NULL pointer!

#### 4.3.3.4 `int Sz_Check_FQDN_or_IPv6_in_RSBL (char * fqdn_or_ipv6, char * rsbl, a_ew_status_t * a_err_war)`

Check if a FQDN or IPv6 is indexed in a Real-Time IPv6 DNS List.

##### Parameters:

*fqdn\_or\_ipv6* pointer to FQDN or IPv6 of the host to verify.

*rsbl* pointer to DNSBL to be consulted.

*a\_err\_war* A pointer to error and warning typedef-structure passed by reference.

##### Note:

If *rsbl* is NULL, default to ip6.rsbl.aupads.org.

##### Returns:

A\_SUCCESS = Host is indexed. The TXT record message is returned in *\*txt\_message\_ptr* and the IP record in DNSBL is returned in *\*ip\_rsbl\_address\_ptr*.

A\_UNSUCCESS = Host is not indexed.

##### Note:

If errors occurred, *antisпам\_error* is set with last error code:

A\_CFI6R\_NULL\_IP\_FQDN = Null IP number passed.

A\_CFI6R\_MEMORY\_ALLOC\_FAIL = Memory alloc fail.

A\_CFI6R\_NO\_HOSTENT\_ASSOCIATED = No hostent associated.

A\_CFI6R\_IP\_NUMBER\_TOO\_SHORT = IP number too short.

A\_CFI6R\_DNS\_TEMP\_FAIL = Temporary DNS fail.

A\_CFI6R\_CREATE\_DNSBL\_QUERY\_FAIL = Create IPv6 DNSBL query fail.

#### 4.3.3.5 `int Sz_Check_Valid_FQDN (char * fqdn, a_boolean_t ipv6, a_ew_status_t * a_err_war)`

Check if a FQDN points to a valid IP number (IPv4 or IPv6).

##### Parameters:

*fqdn* A string pointer to FQDN.

*ipv6* A boolean to define if IP is IPv4 (A\_FALSE) or IPv6 (A\_TRUE).

*a\_err\_war* A pointer to error and warning typedef-structure passed by reference.

##### Returns:

A\_SUCCESS = Points to a valid IP number.

A\_UNSUCCESS = Points to an invalid IP number.

**Note:**

If errors occurred, `antispam_error` is set with last error code:

`A_CVF_NULL_IP_FQDN` = Null FQDN passed.

`A_CVF_ZERO_LEN` = Zero lenght FQDN.

`A_CVF_MEMORY_ALLOC_FAIL` = Memory alloc fail.

`A_CVF_NO_HOSTENT_ASSOCIATED` = No hostent associated.

`A_CVF_IPV4_INVALID_FORMAT` = Invalid IPv4 format.

`A_CVF_IPV6_INVALID_FORMAT` = Invalid IPv6 format.

#### 4.3.3.6 `int Sz_Check_Valid_FQDN_or_IP (char * fqdn_or_ip, a_boolean_t ipv6, a_ew_status_t * a_err_war)`

Check if a FQDN or an IP number (IPv4 or IPv6) is valid.

**Parameters:**

*fqdn\_or\_ip* A string pointer to FQDN or IP number.

*ipv6* A boolean to define if IP to be find is IPv4 (`A_FALSE`) or IPv6 (`A_TRUE`) if string point to an FQDN. Make no sense if string point to an IPv4 or IPv6 number.

*a\_err\_war* A pointer to error and warning typedef-structure passed by reference.

**Returns:**

`A_SUCCESS` = Valid IP or FQDN points to a valid IP number.

`A_UNSUCCESS` = Invalid IP or FQDN points to an invalid IP number.

**Note:**

If errors occurred, `antispam_error` is set with last error code:

`A_CVFI_NULL_IP_FQDN` = Null FQDN or IP number passed.

`A_CVFI_ZERO_LEN` = Zero lenght FQDN.

`A_CVFI_MEMORY_ALLOC_FAIL` = Memory alloc fail.

`A_CVFI_NO_HOSTENT_ASSOCIATED` = No hostent associated.

`A_CVFI_IPV4_INVALID_FORMAT` = Invalid IPv4 format.

`A_CVFI_IPV6_INVALID_FORMAT` = Invalid IPv6 format.

#### 4.3.3.7 `int Sz_Check_Valid_IP_Number (char * ip, a_ew_status_t * a_err_war)`

Check if an IPv4 number belongs to a valid network (non-RFC1918 and others).

##### Parameters:

*ip* A strings pointer to IPv4 number.

*a\_err\_war* A pointer to error and warning typedef-structure passed by reference.

##### Returns:

A\_SUCCESS = Valid IPv4 number.

A\_UNSUCCESS = Invalid IPv4 number.

##### Note:

If errors occurred, `antispam_error` is set with last error code:

A\_CVIN\_OUT\_OF\_RANGE = Octetes out of range.

A\_CVIN\_NULL\_IP\_FQDN = Null IP number passed.

A\_CVIN\_MEMORY\_ALLOC\_FAIL = Memory alloc fail.

A\_CVIN\_NOT\_DOT\_QUAD\_IP = Not a dotted quad IP number associated.

#### 4.3.3.8 `int Sz_Check_Valid_IPv6_Number (char * ipv6, a_ew_status_t * a_err_war)`

Check if an IPv6 number belongs to a valid Global unicast network (non Unspecified, Loopback, Multicast, Link-local unicast, Site-local unicast).

##### Parameters:

*ipv6* A strings pointer to IPv6 number.

*a\_err\_war* A pointer to error and warning typedef-structure passed by reference.

##### Returns:

A\_SUCCESS = Valid IPv6 number (Global unicast)

A\_UNSUCCESS = Invalid IPv6 number (others) or errors (and `antispam_error` is set)

##### Note:

If errors occurred, `antispam_error` is set with last error code:

A\_CVIPV6N\_OOMM = Out of memory.

A\_CVIPV6N\_NULL\_IP = Null IPv6 number passed.

A\_CVIPV6N\_INVALID\_FORMAT = Invalid IPv6 format.

##### Note:

See: RFC-3513 for more details

## 4.4 antispam\_error.h File Reference

LibAntispam Global Errors and Warning Variables and its Functions.

### Data Structures

- struct `a_ew_status_s`

*Store the error and warnings codes called LibAntispam function. The structure for LibAntispam error and warnings.*

### Typedefs

- typedef `a_ew_status_s a_ew_status_t`

*Store the error and warnings codes called LibAntispam function. The typedef-structure for LibAntispam error and warnings.*

### Enumerations

- enum `Antispam_Errors_Code` {

`NOT_ERROR, A_CVIN_NULL_IP_FQDN, A_CVIN_ZERO_LEN, A_CVIN_NOT_DOT_QUAD_IP,`

`A_CVIN_OUT_OF_RANGE, A_CVIN_MEMORY_ALLOC_FAIL, A_CVIPV6N_OOMM, A_CVIPV6N_NULL_IP,`

`A_CVIPV6N_INVALID_FORMAT, A_CVF_NULL_IP_FQDN, A_CVF_ZERO_LEN, A_CVF_MEMORY_ALLOC_FAIL,`

`A_CVF_NO_HOSTENT_ASSOCIATED, A_CVF_IPV4_INVALID_FORMAT, A_CVF_IPV6_INVALID_FORMAT, A_CVFI_NULL_IP_FQDN,`

`A_CVFI_ZERO_LEN, A_CVFI_MEMORY_ALLOC_FAIL, A_CVFI_NO_HOSTENT_ASSOCIATED, A_CVFI_IPV4_INVALID_FORMAT,`

`A_CVFI_IPV6_INVALID_FORMAT, A_CFIR_NOT_IP_OR_FQDN, A_CFIR_NULL_IP_FQDN, A_CFIR_MEMORY_ALLOC_FAIL,`

`A_CFIR_MEMORY_REALLOC_FAIL, A_CFIR_DNS_TEMP_FAIL, A_CFIR_NO_HOSTENT_ASSOCIATED, A_CFIR_IP_NUMBER_TOO_SHORT,`

`A_CFIR_CREATE_DNSBL_QUERY_FAIL, A_CFI6R_NOT_FQDN_OR_IP, A_CFI6R_NULL_IP_FQDN, A_CFI6R_MEMORY_ALLOC_FAIL,`

`A_CFI6R_DNS_TEMP_FAIL, A_CFI6R_NO_HOSTENT_ASSOCIATED, A_CFI6R_CREATE_DNSBL_QUERY_FAIL, A_CFIMR_NULL_IP_FQDN,`

`A_CFIMR_NULL_RSBL_LIST, A_CFI6MR_NULL_IP_FQDN, A_CFI6MR_NULL_RSBL_LIST, MTA_HI_OOMM,`

`MTA_HI_CC_RUN_DIR, MTA_HI_RAL, MTA_HI_GRAYLCD, MTA_HI_RDSSCLCD,`

MTA\_HI\_DIR, MTA\_HI\_IPNAR, MTA\_HI\_GETHOSTNAME, MTA\_HI\_LGETHOSTBYNAME,  
 MTA\_HC\_RNL, MTA\_HSR\_CBCDBUA, MTA\_HSR\_MACFTSZERO, MTA\_HSR\_AHFTSZERO,  
 MTA\_HSR\_PHFTSZERO, MTA\_HSR\_RFSFTSZERO, MTA\_HSR\_BMDFTSZERO, MTA\_HSR\_GMAFTSZERO,  
 MTA\_HSR\_ICGMCFT, MTA\_HSR\_ICGAHFT, MTA\_HSR\_ICGPHFT, MTA\_HSR\_ICGRFSFT,  
 MTA\_HSR\_ICGBMDFT, MTA\_HSR\_ICGGMFT, MTA\_HR\_OOMM, MTA\_HR\_RNL,  
 MTA\_HR\_CUTSZERO, MTA\_HR\_ICGCUT, MTA\_HR\_ICCCULF, MTA\_HCIPN\_OOMM,  
 MTA\_HCIPN\_NULLBIP, MTA\_HCIPN\_NULLBNET, MTA\_HCIPN\_NoZIN, MTA\_CIIA\_OOMM,  
 MTA\_CIIA\_NRDIFRSC, MTA\_CIIA\_NMA, MTA\_CIIA\_BFMA, MTA\_HCRD\_OOMM,  
 MTA\_HCRD\_MDIN, MTA\_HCRD\_NoZIN, MTA\_HCRD\_CTUN, MTA\_HSIRL\_OOMM,  
 MTA\_HSIRL\_NRDIFRSC, MTA\_HSIAHL\_OOMM, MTA\_HSIAHL\_NRDIFRSC, MTA\_HSICUL\_OOMM,  
 MTA\_HSICUL\_NRDIFRSC, MTA\_HSICUL\_NRUI, MTA\_HSICUL\_IEDF, MTA\_HSICUL\_ICARRH,  
 MTA\_HSICUL\_ICARST, MTA\_HSICUL\_EMDL, MTA\_HSICUL\_GETHOSTNAME, MTA\_HSICUL\_LGETHOSTBYNAME,  
 MTA\_HCFMOFIN\_OOMM, MTA\_HCFMOFIN\_EMINL, MTA\_HCFMOFIN\_NRDIFRSC, MTA\_HSC\_OOMM,  
 MTA\_HSC\_NCIP, MTA\_HSC\_NP, MTA\_HSC\_IHI, MTA\_HCMF\_OOMM,  
 MTA\_HCMF\_EMDOMINL, MTA\_HCMF\_NRDIFRSC, MTA\_HCMF\_NSIFMFC, MTA\_HCMF\_NHIFHC,  
 MTA\_HCMF\_UOSBPBR, MTA\_HCMF\_IEDF, MTA\_HCMF\_ICARRH, MTA\_HCMF\_ICARST,  
 MTA\_HCGMF\_OOMM, MTA\_HCGMF\_NRDIFRSC, MTA\_HCGMF\_NSIFMFC, MTA\_HCGMF\_UOSBPBR,  
 MTA\_HCGMF\_IEDF, MTA\_HCGMF\_ICARRH, MTA\_HCGMF\_ICARST, MTA\_HGL\_OOMM,  
 MTA\_HGL\_NRDIFRSC, MTA\_HGL\_NSIFMFC, MTA\_HGL\_NRIFRTC, MTA\_HGL\_GCDNE,  
 MTA\_HGL\_GCDND, MTA\_HGL\_GLDNE, MTA\_HGL\_GLDND, MTA\_HGL\_CGLFF,  
 MTA\_HGL\_CGLCF, MTA\_HGL\_OGLCF, MTA\_HGL\_CTGLCF, MTA\_HGL\_RTGLCF,  
 MTA\_HGLC\_CGLPLFF, MTA\_HBL\_OOMM, MTA\_HBL\_NRDIFSC, MTA\_HCLUS\_OOMM,  
 MTA\_HCLUS\_NRDIFRSC, MTA\_HCLUS\_NRIFRTMFC, MTA\_HCLUS\_EMDL, MTA\_HCLUS\_NRPAIRTMFC,

MTA\_HCLUS\_IEDF, MTA\_HCLUS\_ICARRH, MTA\_HCLUS\_ICARST,  
MTA\_HCLUS\_GETHOSTNAME,  
MTA\_HCLUS\_LGETHOSTBYNAME, MTA\_HCFPIN\_OOMM, MTA\_-  
HCFPIN\_NRDIFRSC, MTA\_HSM\_OOMM,  
MTA\_HSM\_NRDIFRSC, MTA\_HSM\_NSIFMFC, MTA\_HSM\_NRIFRTC,  
MTA\_HSM\_NWM,  
MTA\_HSM\_IMT, MTA\_HSM\_NRPAIRT, MTA\_HSM\_EMDL, MTA\_-  
HSM\_IEDF,  
MTA\_HSM\_FTOPP, MTA\_HSM\_FTCPP, MTA\_HSM\_ICARRH, MTA\_-  
HSM\_ICARST,  
MTA\_HHC\_OOMM, MTA\_HHC\_NRDIFRSC, MTA\_HHC\_NHIFHC,  
MTA\_HHC\_EMDL,  
MTA\_HCRSSDV\_OOMM, MTA\_HCRSSDV\_GETHOSTNAME, MTA\_-  
HCRSSDV\_LGETHOSTBYNAME2, MTA\_HCRSSDV\_SOCKET,  
MTA\_HCRSSDV\_SETSOCKETOPT, MTA\_HCRSSDV\_SELECT, MTA\_-  
HCRSSDV\_GETSOCKETOPT, MTA\_HCRSSDV\_READ,  
MTA\_HCRSSDV\_WRITE, MTA\_HCRSS\_OOMM, MTA\_HCRSS\_-  
NRDIFRSC, MTA\_HCRSS\_NMFIFMFC,  
MTA\_HCRSS\_EMDL, MTA\_HCRSS\_IEDF, MTA\_HCRSS\_ICARRH,  
MTA\_HCRSS\_ICARST,  
MTA\_HCRSS\_IAF, MTA\_HCRSS\_RCDNE, MTA\_HCRSS\_RCDND,  
MTA\_HCRSS\_RLDNE,  
MTA\_HCRSS\_RLDND, MTA\_HVFUDSD\_OOMM, MTA\_HVFUDSD\_-  
NMFRT, MTA\_HVFUDSD\_IEDF,  
MTA\_HVFUDSD\_ICARRH, MTA\_HVFUDSD\_ICARST, MTA\_-  
HVFUDSD\_GETHOSTNAME, MTA\_HVFUDSD\_LGETHOSTBYNAME,  
MTA\_HESC\_OOMM, MTA\_HESC\_NCS, MTA\_HESC\_NRD, MTA\_-  
HESC\_NRDIFRSC,  
MTA\_HESC\_NCIP, MTA\_HESC\_NMF, IPU\_FSNU\_OOMM, IPU\_CIN\_-  
OOMM,  
IPU\_CIN\_NIP, IPU\_CIN\_NNET, IPU\_CIN\_NCIDR, IPU\_CIN\_-  
BADNET,  
IPU\_CIN\_BADCIDR, IPU\_CDI4QE\_OOMM, IPU\_CDI4QE\_NULLS\_-  
AS\_DNSBL\_RS, IPU\_CDI4\_OOMM,  
IPU\_CDI4\_NULLS\_AS\_RS, IP6U\_CTUADDR\_OOMM, IP6U\_-  
CTUADDR\_MT2CIANA, IP6U\_CTFUADDR\_OOMM,  
IP6U\_CTFUADDR\_NULLS\_AS\_RS, IP6U\_CTFUADDR\_GAEINV,  
IP6U\_CTFUADDR\_GAINV, IP6U\_CDI6QE\_OOMM,  
IP6U\_CDI6QE\_NULLS\_AS\_DNSBL\_RS, IP6U\_CDI6QE\_ICTUAE,  
IP6U\_CDI6QE\_GAEINV, IP6U\_CDI6QE\_GAINV,  
IP6U\_RLZACFI6\_OOMM, IP6U\_RLZACFI6\_NULL, IP6U\_CDI6\_OOMM,  
IP6U\_CDI6\_NULLS\_AS\_RS,  
IP6U\_CDI6\_ICTUAE, IP6U\_CDI6\_GAEINV, IP6U\_CDI6\_GAINV, RU\_-  
STSRR\_DEE,  
RU\_STSRR\_NTSRR, RU\_STSRR\_CMPIE, RU\_STSRR\_CMTOOL, RU\_-  
SRRR\_RSE,

```

RU_SRRR_DSNE, RU_SRRR_DEE, RU_SBMXRR_RSE, RU_GMXL_OOMM,
RU_GMXL_RSE, RU_CIPMXRR_OOMM, RU_CIPMXRR_RSE, RU_SCMXRR_OOMM,
RU_SCMXRR_RSE, RU_GRD_OOMM, RU_GRD_MDIN, RU_GRD_NoZIN,
RU_GRD_CTUN, RU_COMPTRRR_RSE, RU_COMGETPTRRR_RSE,
SU_A_STRLWR_PSN,
SU_DECOMMENT_OOMM, SU_DECOMMENT_ARGNULL, SU_REMOVE_SPACE_OOMM,
SU_REMOVE_SPACE_INPUT_STRING_NULL,
SU_CARRH_OOMM, SU_CARRH_PSN, SU_CARST_OOMM, SU_CARST_PSN,
SU_CNS_OOMM, SU_CNS_NRDNSS, SU_CNS_SNTLOW, SU_CNS_NSC,
SU_CNS_STS, SU_CNG_NRDNSS, SU_CNG_DNTLOW, SU_CNG_STS,
SU_CIP4IF_OOMM, SU_CIP4IF_NRDNSS, SU_CIP4IF_NIP4S, SU_CIP6IF_OOMM,
SU_CIP6IF_NRDNSS, SU_CIP6IF_NIP6S, SU_CUCBE_OOMM, SU_CUCBE_NMA,
SU_CUCBE_IEDF, SPF_TXT2STRUCT_OOMM, SPF_TXT2STRUCT_NSTRR,
SPF_TXT2STRUCT_ISTRR,
SPF_TXT2STRUCT_NMF, SPF_TXT2STRUCT_NH, SPF_TXT2STRUCT_NIP,
SPF_TXT2STRUCT_BADMF,
SPF_A_OOMM, SPF_MX_OOMM, SPF_PTR_OOMM, SPF_REDIRECT_OOMM,
SPF_REDIRECT_NSS, SPF_REDIRECT_PARENT_NSS, SPF_REDIRECT_ISTRR,
SPF_REDIRECT_MEE,
SPF_EXP_OOMM, SPF_EXP_MEE, SPF_DSC_OOMM, SPF_DSC_NMF,
SPF_DSC_BADMF, SPF_DSC_NH, SPF_DSC_NRCI, SPF_DSC_NSS,
SPF_MACRO_SME_OOMM, SPF_MACRO_SLPME_OOMM, SPF_MACRO_SLPME_NLPA,
SPF_MACRO_SDME_OOMM,
SPF_MACRO_SDME_NDA, SPF_MACRO_IPME_OOMM, SPF_MACRO_VIPDME_OOMM,
SPF_MACRO_HME_OOMM,
SPF_MACRO_BE_OOMM, SPF_MACRO_MEP_OOMM, SPF_MACRO_MEP_UOCB,
SPF_MACRO_MEP_SYNTAX,
SPF_MACRO_MEP_ESTOOL, SPF_MACRO_SPFME_NO_SPF,
NUMBER_ERRORS }

```

*Antispam return errors code enumeration. Get complete error message for each code using `str_antispam_error()`(p.54).*

- enum Antispam\_Warnings\_Code {

```

NOT_WARNING, MTA_HI_MMDNSBLV4, MTA_HI_MMDNSBLV6,
MTA_HI_MMDS,
MTA_HI_MMMIPNS, MTA_HI_MMMFIPNS, MTA_HI_MMMPIPNS,
MTA_HI_MMWBHDS,

```

```

MTA_HI_MMWBMFAS, MTA_HI_MMWGMFAS, MTA_HI_MMCS,
MTA_HI_MMWS,
MTA_HI_ICOHWRL, MTA_HI_MMWAS, MTA_HI_ICGMACT, MTA_
HI_ICGRFST,
MTA_HI_ICGAHLT, MTA_HI_ICGPHLT, MTA_HI_ICGBMADLT,
MTA_HI_ICGGMALT,
MTA_HI_ICGCUT, MTA_HI_ICFHAHL, MTA_HI_ICFHPHL, MTA_
HI_ICFHBMADL,
MTA_HI_ICFHGMAL, MTA_HI_MMCUSZR, MTA_HI_ICFHAUL,
MTA_HI_NONROOT,
MTA_HSR_NTRTRUE, MTA_HR_NTRTRUE, MTA_HR_HSR, MTA_
HR_ICFCUL,
MTA_HR_MMCUSZR, MTA_HCIPN_PUSEICIPN, MTA_HCIPN_
IEICIPN, MTA_HCIPN_NoZNN,
MTA_CIIA_RFC_1918_MMIPNETOF, MTA_HCRD_MDISTL, MTA_
HSIRL_ERL, MTA_HSIRL_MDISTL,
MTA_HSIAHL_EAHL, MTA_HSIAHL_MDISTL, MTA_HSICUL_EAUL,
MTA_HSICUL_NRPA,
MTA_HCFMOFIN_MDISTL, MTA_HCMF_MFSTL, MTA_HCMF_
IPV4_RFC1918, MTA_HCGMF_WAF,
MTA_HCGMF_MFSTL, MTA_HGL_MDISTL, MTA_HGL_MFUSTL,
MTA_HGL_RCUSTL,
MTA_HGLC_GSGLCFF, MTA_HGLC_RTGLCFF, MTA_HGLC_
RGLCFF, MTA_HGLC_CCTGLCF,
MTA_HBL_MDISTL, MTA_HBL_DI4LN, MTA_HBL_DI6LN, MTA_
HCFPIN_MDISTL,
MTA_HSM_RRINILD, MTA_HHC_IPV4_RFC1918, MTA_HCRSSDV_
CONNECT_FAIL, MTA_HCRSSDV_RECEIVE_TIMEOUT,
MTA_HCRSSDV_SEND_TIMEOUT, MTA_HCRSSDV_RCPT_INVALID,
MTA_HCRSS_MFINRP, MTA_HCRSS_MFSTL,
MTA_HCRSS_MWIPUCL, MTA_HVFUDSD_MFSTL, MTA_
HVFUDSD_RTSTL, IPU_FSNU_NOOR,
IPU_CIN_CCIP6IP4, IPU_CIN_CCIP4IP6, RU_SRRR_RSRTL, RU_
GRD_MDISTL,
SU_CNS_FOUND_NUMERIC_SETS, SU_CNG_FOUND_NUMERIC_
GROUP, SU_CIP4IF_FOUND_IP4_IN_FQDN, SU_CIP4IF_FOUND_
IP4_HEX_IN_FQDN,
SU_CIP6IF_FOUND_IP6_COMPRESS_IN_FQDN, SU_CIP6IF_
FOUND_IP6_UNCOMPRESS_IN_FQDN, SU_CUCBE_FOUND_
UNWANTED_CHAR, SPF_REDIRECT_NO,
SPF_REDIRECT_MAX, SPF_MACRO_SPFME_NO_MACROS, SPF_
MACRO_SPFME_MEE, NUMBER_WARNINGS }

```

*Antispam return warnings code. Get complete warning message for each code using `str_antispam_warning()`(p. 54).*

## Functions

- `char * str_antispam_error (a_ew_status_t a_err_war)`  
*Return a pointer to LibAntispam internal error message.*
- `char * str_antispam_warning (a_ew_status_t a_err_war)`  
*Return a pointer to LibAntispam internal warning message.*

### 4.4.1 Detailed Description

LibAntispam Global Errors and Warning Variables and its Functions.

**Author:**

Rafael Jorge Csura Szendrodi <szendro@pads.ufrj.br>

**Date:**

07/09/2004 GNU Lesser General Public License version 3

### 4.4.2 Function Documentation

#### 4.4.2.1 `char * str_antispam_error (a_ew_status_t a_err_war)`

Return a pointer to LibAntispam internal error message.

**Parameters:**

*a\_err\_war* The typedef-structure for LibAntispam error and warnings.

**Returns:**

The error message associated to error number in typedef-structure or unknown error message if error number is out of range.

#### 4.4.2.2 `char * str_antispam_warning (a_ew_status_t a_err_war)`

Return a pointer to LibAntispam internal warning message.

**Parameters:**

*a\_err\_war* The typedef-structure for LibAntispam error and warnings.

**Returns:**

The warning message associated to warning number in typedef-structure or unknown warning message if warning number is out of range.

## 4.5 mta-heuristics++ File Reference

For compatibility with default use of namespace by GCC-3.x series.

```
#include <mta-heuristics++.h>
```

### 4.5.1 Detailed Description

For compatibility with default use of namespace by GCC-3.x series.

## 4.6 mta-heuristics++.h File Reference

Antispam heuristics API Class (Wild West Antispam Rules) for MTAs.

```
#include <mta-heuristics.h>
```

### Data Structures

- class **SzAntispamMTA**

*Antispam MTA Class front-end to Antispam MTA heuristics functions.*

### 4.6.1 Detailed Description

Antispam heuristics API Class (Wild West Antispam Rules) for MTAs.

**Author:**

Rafael Jorge Csura Szendrodi <szendro@pads.ufrj.br>

**Date:**

24/06/2004

## 4.7 mta-heuristics.h File Reference

Antispam heuristics API routines (Wild West Antispam Rules) for MTAs.

```
#include <antispam_error.h>
```

```
#include <inttypes.h>
```

### Data Structures

- struct **multi\_dnsbl\_s**  
*Store the results for a multi-dnsbl search. The structure for antispam heuristics mta multi\_dnsbl search result.*
- struct **v\_mail\_from\_s**  
*Store the results for a mail from ckeck. The structure for antispam heuristics mta mail from check result.*
- struct **prohibited\_ip\_net\_r\_s**  
*Store the results for check prohibited IP and Networks. The structure for antispam heuristics check prohibited IP/Networks result.*
- struct **v\_helo\_r\_s**  
*Store the results for a bad helo check. The structure for antispam heuristics mta helo check result.*
- struct **control\_user\_search\_r\_s**  
*Store the results for a search in control-users list. The structure for antispam heuristics for a search in control-users list.*

### Defines

- #define **NULL\_DNS\_INFORMATION** "NO-DNS-INFORMATION"  
*Constant string to NULL reverse DNS information.*
- #define **SPECIAL\_COMMAND\_RETURN** "SPECIAL-COMMAND"  
*Constant string to sucessful's special commands execution return.*
- #define **MTA\_USER\_LOCAL\_ERROR** 0x00000001  
*Internal error occurred.*
- #define **MTA\_USER\_LOCAL\_VRFY\_DISABLED** 0x00000002  
*Administrator disabled verification.*
- #define **MTA\_USER\_LOCAL\_NOT** 0x00000004  
*User is not a local user.*
- #define **MTA\_USER\_LOCAL\_INVALID** 0x00000008  
*User is invalid.*
- #define **MTA\_USER\_LOCAL\_RESTRICT** 0x00000010

*User is restrict.*

- `#define MTA_USER_LOCAL_RESTRICT_POST 0x00000020`  
*User is a restricted post mail user (cannot send messages).*
- `#define MTA_USER_LOCAL_IP_ADMIN_REJECT_IP_IN_ADDRESS 0x00000040`  
*Administrator rejects IP numbers in addresses.*
- `#define MTA_USER_LOCAL_IP_NOT_IN_LITERAL_FORMAT 0x00000080`  
*IP number was passed in non-literal format.*
- `#define MTA_USER_LOCAL_IP_FORMAT_INVALID 0x00000100`  
*IP number is not in a valid format.*
- `#define MTA_USER_LOCAL_IP_INVALID_IPV4 0x00000200`  
*The passed IPv4 number is invalid.*
- `#define MTA_USER_LOCAL_IP_INVALID_IPV4_RFC1918 0x00000400`  
*The passed IPv4 number belongs to a private network (RFC-1918).*
- `#define MTA_USER_LOCAL_IP_INVALID_IPV6 0x00000800`  
*The passed IPv6 number is invalid.*
- `#define MTA_USER_LOCAL_IP_IPV6_ADDR_IN_IPV4_CONNECTION 0x00001000`  
*Passed an IPv6 number but the current connection type is IPv4.*
- `#define MTA_USER_LOCAL_IP_IPV4_ADDR_IN_IPV6_CONNECTION 0x00002000`  
*Passed an IPv4 number but the current connection type is IPv6.*
- `#define MTA_USER_LOCAL_IP_NOT_RESOLVE_TO_FQDN 0x00004000`  
*Passed IP number don't resolve to a Fully Qualified Domain Name.*
- `#define V_NONE 0x0000000000000001LL`  
*No check will be procced. User search found user and no check will be procced.*
- `#define V_ERROR 0x0000000000000002LL`  
*Search returned errors. User search return on errors (can be considered a synonym to V\_NO\_USER).*
- `#define V_NO_USER 0x0000000000000004LL`  
*No found user. User search no found user in controled users list.*
- `#define V_INVALID 0x0000000000000008LL`  
*Invalid user. User search found user and he is an invalid user.*
- `#define V_RESTRICT 0x0000000000000010LL`

*Restricted user. User search found user and he is a restricted user.*

- **#define V\_RESTRICT\_POST** 0x0000000000000020LL  
*User has restricted post rights. User search found user and he has restricted post rights.*
- **#define V\_WARNING\_INSTEAD\_REJECT** 0x0000000000000040LL  
*Don't reject messages, warning instead. User search found user and he don't want to reject messages, but add a warning in the subject of message (not implemented yet).*
- **#define V\_NOTIFY** 0x0000000000000080LL  
*Notify recipient user. User search found user and he want to receive a mail warning if a mail is refused by any rules below, except Graylist, and Invalid or Restricted user above. This return value take no sense without one or more rules enabled.*
- **#define V\_REVERSE** 0x0000000000000100LL  
*Check reverse DNS. User search found user and he want to check reverse DNS for remote host.*
- **#define V\_PROHIBITED\_FQDN\_STRINGS** 0x0000000000000200LL  
*Check prohibited FQDN strings. User search found user and he want to check if remote host has a prohibited FQDN string in reverse DNS (typically dialup strings), if reverse DNS exist.*
- **#define V\_BLACKLIST\_NO** 0x0000000000000400LL  
*No use blacklists. User search found user and he don't want to use blacklist.*
- **#define V\_BLACKLIST\_YES** 0x0000000000000800LL  
*Use blacklists. User search found user and he want to use blacklist.*
- **#define V\_BLACKLIST\_PERSIST** 0x0000000000001000LL  
*Use blacklists in persistent mode. User search found user and he want to USE blacklist in PERSISTENT mode (refuse mail while one or more blacklist(s) return TRY\_AGAIN (temporary DNS error)).*
- **#define V\_BLACKLIST\_ADMIN\_DECIDES** 0x0000000000002000LL  
*Use blacklists only if administrator choose it. User search found user and he want to USE blacklist in ADMIN\_DECIDES mode (lets system administrator choose if use blacklist with/without PERSISTENT mode or not use blacklist).*
- **#define V\_GRAYLIST** 0x0000000000004000LL  
*Use graylist. User search found user and he want to use graylist.*
- **#define V\_WHITELIST\_SPF\_NO** 0x0000000000008000LL  
*No use whitelist SPF-like. User search found user and he don't want to use whitelist (based on SPF - Sender Policy Framework) (not implemented yet, only for LibAntispam 1.3.0 or up).*
- **#define V\_WHITELIST\_SPF\_YES** 0x0000000000010000LL  
*Use whitelist SPF-like. User search found user and he want to use whitelist (not implemented yet, only for LibAntispam 1.3.0 or up).*
- **#define V\_WHITELIST\_SPF\_PERSIST** 0x0000000000020000LL  
*Use whitelist SPF-like in persistent mode. User search found user and he want to USE whitelist in PERSISTENT mode (refuse mail while whitelist return TRY\_AGAIN (temporary DNS error)) (not implemented yet, only for LibAntispam 1.3.0 or up).*

- **#define V\_WHITELIST\_SPF\_ADMIN\_DECIDES** 0x000000000040000LL  
*Use whitelist SPF-like only if administrator choose it. User search found user and he want to USE whitelist in ADMIN\_DECIDES mode (lets system administrator choose if use whitelist with/without PERSISTENT mode or not use whitelist) (not implemented yet, only for LibAntispam 1.3.0 or up).*
- **#define V\_BAD\_HELO** 0x000000000080000LL  
*Verify for bad SMTP HELO command. User search found user and he want to restrict bad HELO command use.*
- **#define V\_REMOTE\_DOMAIN\_SMTP\_SENDER\_CHECKING** 0x0000000000100000LL  
*Verify SMTP connection in remote domain or MX for sender address. User search found user and he want to verify if remote domain or its MX accept SMTP connections for address passed in MAIL FROM command.*
- **#define V\_FROM** 0x000000000020000LL  
*Verify SMTP MAIL FROM command. User search found user and he want to verify MAIL FROM command passed by remote host.*
- **#define V\_USER\_RESTRICT\_NULL\_REVERSE\_PATH\_USE** 0x0000000000400000LL  
*Restrict NULL reverse-path. In V\_FROM, User search found user and he want to restrict NULL reverse-path "<>" use.*
- **#define V\_USER\_RESTRICT\_SENDER\_USING\_SPF** 0x0000000000800000LL  
*Use SPF to check MAIL FROM command. In V\_FROM, User search found user and he want to use SPF.*
- **#define V\_USER\_RESTRICT\_LOCAL\_DOMAIN\_USE** 0x00000000001000000LL  
*Restrict local domain use. In V\_FROM, User search found user and he want to restrict his local domain use.*
- **#define V\_USER\_RESTRICT\_PROHIBITED\_CHARS\_USE** 0x00000000002000000LL  
*Restrict for prohibited characteres. In V\_FROM, User search found user and he want to restrict prohibited characteres use ( default prohibited characteres are |,\,\_, ~, ', !, #, \$, ^, &, \*, (, ), {, }, [, ], ", ', ;, :, ;, ?, / ).*
- **#define V\_USER\_RESTRICT\_BAD\_MAIL\_FROM\_USE** 0x00000000004000000LL  
*Restrict bad address in SMTP MAIL FROM use. In V\_FROM, User search found user and he want to restrict bad MAIL FROM command use.*
- **#define V\_PROHIBITED\_IP\_NETWORKS** 0x00000000008000000LL  
*Use basic IP/Network deny list. User search found user and he want to use a basic IP/Network restriction (deny list).*
- **#define V\_HEADER\_ANALYZE** 0x000000000010000000LL  
*Proceed a real-time header analyze. User search found user and he want to proceed a real-time header analyze (not implemented yet, only for future versions of LibAntispam).*

- `#define V_SENDER_AUTH 0x0000000020000000LL`  
*Sender was authenticated by any authentication mechanism This flag is set if the sender was authenticated by any authentication mechanism.*
- `#define V_RCPT_AUTH 0x0000000040000000LL`  
*Recipient was authenticated by any authentication mechanism This flag is set if the recipient was authenticated by any authentication mechanism.*
- `#define V_SENDER_AUTH_DONT_DISABLE_GRAYLIST 0x0000000080000000LL`  
*Sender was authenticated but GrayList is not disabled This flag is set if the sender was authenticated but GrayList is wanted.*
- `#define V_SENDER_AUTH_DONT_DISABLE_RDSSC 0x0000000100000000LL`  
*Sender was authenticated but RDSSC is not disabled This flag is set if the sender was authenticated but RDSSC is wanted.*
- `#define V_SENDER_RDSSC_GREETINGS_FAIL 0x0000000200000000LL`  
*The Remote Domain SMTP Sender Checking (RDSSC) failed in greetings The Remote Domain SMTP Sender Checking failed due a temporarily deferred error in SMTP greetings.*
- `#define V_SENDER_RDSSC_TRANSIENT_FAIL 0x0000000400000000LL`  
*The Remote Domain SMTP Sender Checking (RDSSC) failed due transient failure The Remote Domain SMTP Sender Checking failed due a transient error in SMTP transaction.*
- `#define V_SENDER_RDSSC_PERMANENT_FAIL 0x0000000800000000LL`  
*The Remote Domain SMTP Sender Checking (RDSSC) failed due permanent failure The Remote Domain SMTP Sender Checking failed due a permanent error in SMTP transaction.*

## Typedefs

- `typedef multi_dnsbl_s multi_dnsbl_t`  
*Store the results for a multi-dnsbl search. The typedef-structure for antispam heuristics mta multi\_dnsbl search result.*
- `typedef v_mail_from_s v_mail_from_t`  
*Store the results for a mail from check. The typedef-structure for antispam heuristics mta mail from check result.*
- `typedef prohibited_ip_net_r_s prohibited_ip_net_r_t`  
*Store the results for check prohibited IP and Networks. The typedef-structure for antispam heuristics check prohibited IP/Networks result.*
- `typedef v_helo_r_s v_helo_r_t`  
*Store the results for a bad helo ckeck. The typedef-structure for antispam heuristics mta helo check result.*
- `typedef control_user_search_r_s control_user_search_r_t`

*Store the results for a search in control-users list. The typedef-structure for antispam heuristics for a search in control-users list.*

## Enumerations

- enum `a_boolean_t` { `A_FALSE` = 0, `A_TRUE` = 1 }
  - enum `dns_error_t` {  
`DNS_CHECK_DISABLED` = 0, `DNS_CHECK_OK` = 1, `DNS_CHECK_FAIL` = 2, `DNS_CHECK_TEMP_FAIL` = 3,  
`DNS_CHECK_FORGED` = 4 }
- Antispam heuristics dns\_error\_t type definition.*
- enum `dnsbl_r_t` { `DNSBL_NOT` = 0, `DNSBL_IN` = 1, `DNSBL_TEMP_FAIL` = 2, `DNSBL_ERROR` = 3 }
- Antispam heuristics dnsbl result type definition.*
- enum `Sendmail_Message_Types` { `MESSAGE_NONE`, `MESSAGE_RCPT_USER_WARNING`, `NUMBER_OF_MESSAGES` }
- The message types for Sz\_MTA\_Heuristics\_Sendmail()(p. 83).*
- enum `v_mail_from_return_t` {  
`V_MAIL_FROM_ERROR` = 0, `V_MAIL_FROM_OK` = 1, `V_MAIL_FROM_CHECK_IP_ADMIN_REJECT_IP_IN_ADDRESS` = 2, `V_MAIL_FROM_CHECK_IP_NOT_IN_LITERAL_FORMAT` = 3,  
`V_MAIL_FROM_CHECK_IP_FORMAT_INVALID` = 4, `V_MAIL_FROM_CHECK_IP_INVALID_IPV4` = 5, `V_MAIL_FROM_CHECK_IP_INVALID_IPV4_RFC1918` = 6, `V_MAIL_FROM_CHECK_IP_INVALID_IPV6` = 7,  
`V_MAIL_FROM_CHECK_IP_IPV6_ADDR_IN_IPV4_CONNECTION` = 8, `V_MAIL_FROM_CHECK_IP_IPV4_ADDR_IN_IPV6_CONNECTION` = 9, `V_MAIL_FROM_CHECK_IP_NOT_RESOLVE_TO_FQDN` = 10, `V_MAIL_FROM_NULL_REVERSE_PATH_NOT_LOCAL` = 11,  
`V_MAIL_FROM_LOCAL_BUT_CLIENT_NOT_LOCAL` = 12, `V_MAIL_FROM_MATCH_MY_DOMAINS` = 12, `V_MAIL_FROM_RESOLVE_INVALID_IPV4` = 13, `V_MAIL_FROM_RESOLVE_INVALID_IPV6` = 14,  
`V_MAIL_FROM_RESOLVE_MATCH_MY_IP_NETWORKS` = 15, `V_MAIL_FROM_MX_MATCH_MY_DOMAINS` = 16, `V_MAIL_FROM_RESOLVE_MX_INVALID_IPV4` = 17, `V_MAIL_FROM_RESOLVE_MX_INVALID_IPV4_RFC1918` = 18,  
`V_MAIL_FROM_RESOLVE_MX_INVALID_IPV6` = 19, `V_MAIL_FROM_MX_RESOLVE_MATCH_MY_IP_NETWORKS` = 20, `V_MAIL_FROM_DNS_TEMP_FAIL` = 21, `V_MAIL_FROM_NOT_RESOLVE_AND_NOT_HAS_MX` = 22,  
`V_MAIL_FROM_RESOLVE_INVALID_IPV4_RFC1918_AND_NOT_HAS_MX` = 23, `V_MAIL_FROM_BAD_MX_RECORD` = 24, `V_MAIL_FROM_HAS_PROHIBITED_CHARS` = 25, `V_MAIL_FROM_HAS_BAD_ADDRESS` = 26,

```
V_MAIL_FROM_HAS_UNWANTED_CHAR_IN_BEGIN_OR_END -
USERNAME = 27, V_MAIL_FROM_SPF_NONE = 28, V_MAIL_FROM -
SPF_PASS = 29, V_MAIL_FROM_SPF_FAIL = 30,
```

```
V_MAIL_FROM_SPF_SOFTFAIL = 31, V_MAIL_FROM_SPF -
NEUTRAL = 32, V_MAIL_FROM_SPF_TEMPERROR = 33, V_MAIL -
FROM_SPF_PERMERROR = 34 }
```

*Antispam heuristics v\_mail\_from\_return\_t type definition. Antispam heuristics return values for MAIL FROM command check in Sz\_MTA\_Heuristics\_Check-Mail\_From()(p.73).*

- enum `helo_error_t` {

```
HELO_ERROR = 0, HELO_OK = 1, HELO_ARGS_REQUIRED = 2, HELO -
IP_NOT_LITERAL = 3,
```

```
HELO_IP_FORMAT_INVALID = 4, HELO_IPV4_RFC1918 = 5, HELO -
IPV6_HELO_IN_IPV4_CONNECTION = 6, HELO_IPV4_HELO_IN -
IPV6_CONNECTION = 7,
```

```
HELO_ARGS_MY_IP_AND_NETWORKS = 8, HELO_ARGS_MY -
DOMAINS = 9, HELO_DOMAIN_HAS_PROHIBITED_CHARS = 10,
HELO_DOMAIN_HAS_NOT_DOT = 11,
```

```
HELO_DOMAIN_START_WITH_DOT = 12, HELO_DOMAIN_END -
WITH_DOT = 13, HELO_DOMAIN_RESOLVE_INVALID_IPV4 = 14,
HELO_DOMAIN_RESOLVE_INVALID_IPV6 = 15,
```

```
HELO_DOMAIN_NOT_RESOLVE_AND_NOT_HAS_MX = 16, HELO -
DOMAIN_RESOLVE_INVALID_IPV4_RFC1918_AND_NOT_HAS_MX
= 17, HELO_DOMAIN_BAD_MX_RECORD = 18, HELO_DOMAIN -
RESOLVE_MX_INVALID_IPV4 = 19,
```

```
HELO_DOMAIN_RESOLVE_MX_INVALID_IPV4_RFC1918 = 20,
HELO_DOMAIN_RESOLVE_MX_INVALID_IPV6 = 21, HELO -
DOMAIN_RESOLVE_MATCH_MY_IP_NETWORKS = 22, HELO -
DOMAIN_MX_MATCH_MY_DOMAINS = 23,
```

```
HELO_DOMAIN_MX_RESOLVE_MATCH_MY_IP_NETWORKS = 24,
HELO_DOMAIN_DNS_TEMP_FAIL = 25, HELO_HAS_PROHIBITED -
IP_NETWORK = 26, HELO_PROHIBITED_ALL_IPV4_NUMBERS = 27,
```

```
HELO_PROHIBITED_ALL_IPV6_NUMBERS = 28, HELO_HAS -
PROHIBITED_DOMAIN = 29 }
```

*Antispam heuristics helo\_error\_t type definition.*

- enum `graylist_return_t` {

```
GRAYLIST_DISABLED = 0, GRAYLIST_SKIP = 1, GRAYLIST_IN = 2,
GRAYLIST_OUT = 3,
```

```
GRAYLIST_SENDER_AUTH = 4 }
```

*Antispam heuristics graylist\_return\_t type definition.*

- enum `remote_smtp_sender_return_t` {

```
REMOTE_SMTP_SENDER_DISABLED = 0, REMOTE_SMTP -
SENDER_SKIP = 1, REMOTE_SMTP_SENDER_MTA_GREETINGS -
FAIL_GRAYLIST_DELAY = 2, REMOTE_SMTP_SENDER -
PERMANENT_FAIL_GRAYLIST_DELAY = 3,
```

```
REMOTE_SMTP_SENDER_TRANSIENT_FAIL_GRAYLIST_DELAY
= 4, REMOTE_SMTP_SENDER_PERMANENT_FAIL = 5, REMOTE_
SMTP_SENDER_TRANSIENT_FAIL = 6, REMOTE_SMTP_SENDER_
OK = 7,
```

```
REMOTE_SMTP_SENDER_SENDER_AUTH = 8, REMOTE_SMTP_
SENDER_CHECK_IP_ADMIN_REJECT_IP_IN_ADDRESS = 9,
REMOTE_SMTP_SENDER_CHECK_IP_NOT_IN_LITERAL_FORMAT
= 10, REMOTE_SMTP_SENDER_CHECK_IP_FORMAT_INVALID = 11,
REMOTE_SMTP_SENDER_CHECK_IP_INVALID_IPV4 = 12,
REMOTE_SMTP_SENDER_CHECK_IP_INVALID_IPV4_RFC1918
= 13, REMOTE_SMTP_SENDER_CHECK_IP_INVALID_IPV6 = 14,
REMOTE_SMTP_SENDER_CHECK_IP_IPV6_ADDR_IN_IPV4_
CONNECTION = 15,
```

```
REMOTE_SMTP_SENDER_CHECK_IP_IPV4_ADDR_IN_IPV6_
CONNECTION = 16, REMOTE_SMTP_SENDER_CHECK_IP_NOT_
RESOLVE_TO_FQDN = 17 }
```

*Antispam heuristics remote\_smtp\_sender\_return\_t type definition.*

- enum `global_user_search_type_t` { `NOT_USER` = 0, `DEFAULT_USER` = 1, `NORMAL_USER` = 2 }

*Global user search types definition.*

## Functions

- `a_boolean_t Sz_MTA_Heuristics_Init(a_ew_status_t *a_err_war)`  
*Init configs, rejected words and accepted hosts/users lists.*
- `void Sz_MTA_Heuristics_Clear(a_ew_status_t *a_err_war)`  
*Dealloc memory for rejected words and accepted hosts/users lists and for others init configs.*
- `a_boolean_t Sz_MTA_Heuristics_Rebuild(a_ew_status_t *a_err_war)`  
*Rebuild system configurations or control\_user list and private users configurations.*
- `dns_error_t Sz_MTA_Heuristics_Check_Reverse_DNS(char *RFC5321_Remote_Hostname_IP_Information, control_user_search_r_t user_opt, a_ew_status_t *a_err_war)`  
*Check Reverse DNS in agreement with RFC-1912.*
- `char * Sz_MTA_Heuristics_Search_In_Rejected_List(char *RFC5321_Remote_Hostname_IP_Information, control_user_search_r_t user_opt, a_ew_status_t *a_err_war)`  
*Find a domain's substring in rejected words list.*
- `a_boolean_t Sz_MTA_Heuristics_Search_In_Accepted_Hosts_List(char *RFC5321_Remote_Hostname_IP_Information, a_ew_status_t *a_err_war)`  
*Find host in accepted hosts lists.*
- `control_user_search_r_t Sz_MTA_Heuristics_Search_In_Controlled_Users_List(char *RFC5321_Remote_Hostname_IP_Information, char *Rcpt_User, a_ew_status_t *a_err_war)`

*Find user in controlled users lists. Return values below are combined and should be verified using AND (&) operation.*

- **a\_boolean\_t Sz\_MTA\_Heuristics\_Check\_For\_My\_or\_Friend\_IP\_Network** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, control\_user\_search\_r\_t user\_opt, a\_ew\_status\_t \*a\_err\_war)  
*Check if remote client is local or friend IP/Network.*
- **v\_mail\_from\_t Sz\_MTA\_Heuristics\_Check\_Mail\_From** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, char \*Helo, control\_user\_search\_r\_t \*user\_opt, a\_ew\_status\_t \*a\_err\_war)  
*Check for legality of sender user declared in MAIL FROM command.*
- **a\_boolean\_t Sz\_MTA\_Heuristics\_Check\_Good\_Mail\_From** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, control\_user\_search\_r\_t \*user\_opt, a\_ew\_status\_t \*a\_err\_war)  
*Check if sender user declared in MAIL FROM cmd is a good user.*
- **graylist\_return\_t Sz\_MTA\_Heuristics\_GrayList** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, char \*Rcpt\_User, control\_user\_search\_r\_t user\_opt, a\_ew\_status\_t \*a\_err\_war)  
*Use Advanced GrayList support.*
- **a\_boolean\_t Sz\_MTA\_Heuristics\_GrayList\_Clean** (a\_ew\_status\_t \*a\_err\_war)  
*Clean up the GrayList cache directory and its entries.*
- **multi\_dnsbl\_t Sz\_MTA\_Heuristics\_BlackList** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, control\_user\_search\_r\_t user\_opt, a\_ew\_status\_t \*a\_err\_war)  
*Use Advanced BlackList support.*
- **int Sz\_MTA\_Heuristics\_Check\_Local\_User\_Status\_In\_Mail\_From** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*User\_Address, a\_ew\_status\_t \*a\_err\_war)  
*Verify if user address in MAIL FROM cmd is local and if is Invalid, Restricted or Non-Sender.*
- **int Sz\_MTA\_Heuristics\_Check\_Local\_User\_Status\_In\_Rcpt\_To** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*User\_Address, a\_ew\_status\_t \*a\_err\_war)  
*Verify if user address in RCPT TO cmd is local and if is Invalid, Restricted or Non-Sender.*
- **prohibited\_ip\_net\_r\_t Sz\_MTA\_Heuristics\_Check\_For\_Prohibited\_IP\_Network** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, control\_user\_search\_r\_t user\_opt, a\_ew\_status\_t \*a\_err\_war)  
*Check if remote client is/belongs to a prohibited IP/Network.*
- **a\_boolean\_t Sz\_MTA\_Heuristics\_Sendmail** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, char \*Rcpt\_User, char \*message, int message\_type, control\_user\_search\_r\_t user\_opt, a\_ew\_status\_t \*a\_err\_war)  
*Send a warning mail to somebody.*

- **v\_helo\_r\_t Sz\_MTA\_Heuristics\_Helo\_Check** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Helo, **control\_user\_search\_r\_t** user\_opt, **a\_ew\_status\_t** \*a\_err\_war)  
*Verify HELO SMTP command for bad things.*
- **remote\_smtp\_sender\_return\_t Sz\_MTA\_Heuristics\_Check\_Remote\_Smtp\_Sender** (char \*RFC5321\_Remote\_Hostname\_IP\_Information, char \*Mail\_From, **control\_user\_search\_r\_t** user\_opt, **a\_ew\_status\_t** \*a\_err\_war)  
*Verify if domain/MX accept SMTP connection for a sender address.*
- **a\_boolean\_t Sz\_MTA\_Heuristics\_Vrfy\_For\_Sender\_User\_Deny\_Smtp\_Data** (char \*Mail\_From, **a\_ew\_status\_t** \*a\_err\_war)  
*Verify if the user part name of sender information match with a user, in internal list, that is prohibited to do the DATA SMTP command (should be implemented in MTA, but is quit simple, we need only return a 550 that is the permanent error code in SMTP messages when DATA command is used).*
- **a\_boolean\_t Sz\_MTA\_Heuristics\_Vrfy\_For\_Recipient\_User\_Deny\_Smtp\_Data** (char \*Rcpt\_To, **a\_ew\_status\_t** \*a\_err\_war)  
*Verify if the user part name of recipient information match with a user, in internal list, that is prohibited to do the DATA SMTP command (should be implemented in MTA, but is quit simple, we need only return a 451 that is the temporary error code in SMTP messages).*

## Variables

- **char \* alt\_mta\_antispam\_conf**  
*Alternative mta-antispam.conf file. Alternative configuration file for mta-antispam.conf.*
- **char \* alt\_rejected\_fqdn\_str**  
*Alternative reject.FQDN.strings file. Alternative configuration file for reject.FQDN.strings.*
- **char \* alt\_accepted\_hosts**  
*Alternative accepted.hosts file. Alternative configuration file for accepted.hosts.*
- **char \* alt\_prohibited\_hosts**  
*Alternative prohibited.hosts file. Alternative configuration file for prohibited.hosts.*
- **char \* alt\_control\_users**  
*Alternative control.users file. Alternative configuration file for control.users.*
- **char \* alt\_maddr\_or\_dom\_bad**  
*Alternative mail-addresses\_or\_domains.bad file. Alternative configuration file for mail-addresses\_or\_domains.bad.*
- **char \* alt\_maddr\_good**  
*Alternative mail-addresses.good file. Alternative configuration file for mail-addresses.good.*
- **a\_boolean\_t disable\_mta\_heuristics**

*Disable mta-heuristics support. Global variable to disable LibAntispam mta-heuristics support for MTA.*

- **a\_boolean\_t rebuilding\_antispam\_rules**

*Rebuilding antispam rules state. Global variable to sign rebuild state to LibAntispam's MTA application.*

- **char \* graylist\_relay\_deny\_message**

*Customized graylist temporary message. Pointer to a customized grayList temporary relay deny message.*

- **char \* deny\_ip\_network\_ptr**

*Deny IP/Network pointer. Pointer to a denied IP/Network found in list.*

## 4.7.1 Detailed Description

Antispam heuristics API routines (Wild West Antispam Rules) for MTAs.

**Author:**

Rafael Jorge Csura Szendrodi <szendro@pads.ufrj.br>

**Date:**

15/06/2004 GNU Lesser General Public License version 3

## 4.7.2 Typedef Documentation

### 4.7.2.1 control\_user\_search\_r\_t

Store the results for a search in control-users list. The typedef-structure for antispam heuristics for a search in control-users list.

**See also:**

**Sz\_MTA\_Heuristics\_Search\_In\_Controlled\_Users\_List()**(p. 80)

### 4.7.2.2 multi\_dnsbl\_t

Store the results for a multi-dnsbl search. The typedef-structure for antispam heuristics mta multi\_dnsbl search result.

**See also:**

**Sz\_MTA\_Heuristics\_BlackList()**(p. 68)

### 4.7.2.3 prohibited\_ip\_net\_r\_t

Store the results for check prohibited IP and Networks. The typedef-structure for antispam heuristics check prohibited IP/Networks result.

**See also:**

**Sz\_MTA\_Heuristics\_Check\_For\_Prohibited\_IP\_Network()**(p. 70)

#### 4.7.2.4 v\_helo\_r\_t

Store the results for a bad helo ckeck. The typedef-structure for antispam heuristics mta helo check result.

See also:

`Sz_MTA_Heuristics_Helo_Check()`(p. 78)

#### 4.7.2.5 v\_mail\_from\_t

Store the results for a mail from ckeck. The typedef-structure for antispam heuristics mta mail from check result.

See also:

`Sz_MTA_Heuristics_Check_Mail_From()`(p. 73)

### 4.7.3 Enumeration Type Documentation

#### 4.7.3.1 enum v\_mail\_from\_return\_t

Antispam heuristics v\_mail\_from\_return\_t type definition. Antispam heuristics return values for MAIL FROM command check in `Sz_MTA_Heuristics_Check_Mail_From()`(p. 73).

Note:

V\_MAIL\_FROM\_LOCAL\_BUT\_CLIENT\_NOT\_LOCAL is deprecated, use the V\_MAIL\_FROM\_MATCH\_MY\_DOMAINS instead.

### 4.7.4 Function Documentation

#### 4.7.4.1 multi\_dnsbl\_t Sz\_MTA\_Heuristics\_BlackList (char \* *RFC5321\_Remote\_Hostname\_IP\_Information*, control\_user\_search\_r\_t *user\_opt*, a\_ew\_status\_t \* *a\_err\_war*)

Use Advanced BlackList support.

Parameters:

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

Returns:

DNSBL\_IN if IP was was found in one BlackList return message in (multi\_dnsbl\_t).message ptr.

DNSBL\_NOT if IP wasn't found in BlackList or on errors.

DNSBL\_TEMP\_FAIL if one of ours dnsbl return message in (multi\_dnsbl\_t).message ptr.

DNSBL\_ERROR on errors and a\_err\_war->antispam\_error is set.

Note:

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This function is controled by user options!

**See also:**

`Sz_MTA_Heuristics_Search_In_Controlled_Users_List()`(p. 80)

**4.7.4.2** `a_boolean_t Sz_MTA_Heuristics_Check_For_My_or_Friend_IP_Network` (char \* *RFC5321\_Remote\_Hostname\_IP\_Information*, control\_user\_search\_r\_t *user\_opt*, a\_ew\_status\_t \* *a\_err\_war*)

Check if remote client is local or friend IP/Network.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

A\_TRUE if remote client is local or friend IP/Network.

A\_FALSE if not or on errors.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This function is controled by user options!

**See also:**

`Sz_MTA_Heuristics_Search_In_Controlled_Users_List()`(p. 80)

**4.7.4.3 prohibited\_ip\_net\_r\_t Sz\_MTA\_Heuristics\_Check\_For\_Prohibited\_IP\_Network** (char \* *RFC5321\_Remote\_Hostname\_IP\_Information*, control\_user\_search\_r\_t *user\_opt*, a\_ew\_status\_t \* *a\_err\_war*)

Check if remote client is/belongs to a prohibited IP/Network.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

A\_TRUE if remote client is/belongs to a prohibited IP/Network.

**In this case, denied IP/Network is pointed by \*deny\_ip\_network\_ptr pointer.**

**Returns:**

A\_FALSE if not or on errors.

**Note:**

if mta heuristics is disabled, return A\_FALSE always.

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This function is controled by user options!

**See also:**

`Sz_MTA_Heuristics_Search_In_Controlled_Users_List()`(p.80)

**4.7.4.4 a\_boolean\_t Sz\_MTA\_Heuristics\_Check\_Good\_Mail\_From (char \* RFC5321\_Remote\_Hostname\_IP\_Information, char \* Mail\_From, control\_user\_search\_r\_t \* user\_opt, a\_ew\_status\_t \* a\_err\_war)**

Check if sender user declared in MAIL FROM cmd is a good user.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*Mail\_From* A char string pointer to MAIL FROM user information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

A\_TRUE if sender user is a good user.

A\_FALSE if not or on errors (and antispam\_error is set!)

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

Mail\_From SHOULD has the format: user@anything.foo.faa or <user@anything.foo.faa>

**4.7.4.5** int Sz\_MTA\_Heuristics\_Check\_Local\_User\_Status\_In\_Mail\_From (char \* RFC5321\_Remote\_Hostname\_IP\_Information, char \* User\_Address, a\_ew\_status\_t \* a\_err\_war)

Verify if user address in MAIL FROM cmd is local and if is Invalid, Restricted or Non-Sender.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*User\_Address* A char pointer to sender user address.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

MTA\_USER\_LOCAL\_OK if local user is VALID and not RESTRICT.

MTA\_USER\_LOCAL\_INVALID if local user is INVALID.

MTA\_USER\_LOCAL\_RESTRICT if local user is RESTRICT.

MTA\_USER\_LOCAL\_NON\_SENDER if local user is NON-SENDER user.

**Note:**

This function check internally the user\_opt.global\_user\_search\_return\_code variable and user\_opt.global\_user\_found\_in\_reduced\_format for the given mail address, if it belongs to our control.users list.

This fuction is a front-end to internal function Sz\_MTA\_Heuristics\_Check\_User\_Status().

**4.7.4.6** int Sz\_MTA\_Heuristics\_Check\_Local\_User\_Status\_In\_Rcpt\_To (char \* RFC5321\_Remote\_Hostname\_IP\_Information, char \* User\_Address, a\_ew\_status\_t \* a\_err\_war)

Verify if user address in RCPT TO cmd is local and if is Invalid, Restricted or Non-Sender.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*User\_Address* A char pointer to recipient user address.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

MTA\_USER\_LOCAL\_OK if local user is VALID and not RESTRICT.  
 MTA\_USER\_LOCAL\_INVALID if local user is INVALID.  
 MTA\_USER\_LOCAL\_RESTRICT if local user is RESTRICT.  
 MTA\_USER\_LOCAL\_NON\_SENDER if local user is NON-SENDER user.

**Note:**

This function check internally the user\_opt.global\_user\_search\_return\_code variable and user\_opt.global\_user\_found\_in\_reduced\_format for the given mail address, if it belongs to our control.users list.

This fuction is a front-end to internal function Sz\_MTA\_Heuristics\_Check\_User\_Status().

#### 4.7.4.7 v\_mail\_from\_t Sz\_MTA\_Heuristics\_Check\_Mail\_From (char \* RFC5321\_Remote\_Hostname\_IP\_Information, char \* Mail\_From, char \* Helo, control\_user\_search\_r\_t \* user\_opt, a\_ew\_status\_t \* a\_err\_war)

Check for legality of sender user declared in MAIL FROM command.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*Mail\_From* A char string pointer to MAIL FROM user information.

*Helo* A char string pointer to HELO information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

V\_MAIL\_FROM\_OK if no problem found in sender information.

V\_MAIL\_FROM\_NULL\_REVERSE\_PATH\_NOT\_LOCAL if null reverse-path "<>" was declared by sender user, but remote client isn't a local or friend machine.

V\_MAIL\_FROM\_LOCAL\_BUT\_CLIENT\_NOT\_LOCAL if sender is local and remote client isn't a local or friend machine.

V\_MAIL\_FROM\_HAS\_PROHIBITED\_CHARS if sender has prohibited characteres and prohibited\_char variable is set.

V\_MAIL\_FROM\_ERROR on errors.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: IDENT:anbody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

Mail\_From SHOULD has the format: user@anything.foo.faa or <user@anything.foo.faa>  
This function is controled by user options!

**See also:**

Sz\_MTA\_Heuristics\_Search\_In\_Controlled\_Users\_List()(p.80)

**4.7.4.8 remote\_smtp\_sender\_return\_t Sz\_MTA\_Heuristics\_Check\_Remote\_SMTP\_Sender (char \* RFC5321\_Remote\_Hostname\_IP\_Information, char \* Mail\_From, control\_user\_search\_r\_t user\_opt, a\_ew\_status\_t \* a\_err\_war)**

Verify if domain/MX accept SMTP connection for a sender address.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*Mail\_From* A char string pointer to address passed in MAIL FROM command.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

REMOTE\_SMTP\_SENDER\_OK if domain/MX accept sender address.

REMOTE\_SMTP\_SENDER\_TRANSIENT\_FAIL if any fase of SMTP transaction with domain/MX return a transient fail condition.

REMOTE\_SMTP\_SENDER\_PERMANENT\_FAIL if any fase of SMTP transaction with domain/MX return a permanent fail condition.

REMOTE\_SMTP\_SENDER\_MTA\_GREETINGS\_FAIL\_GRAYLIST\_DELAY if in greetings fase SMTP transaction with domain/MX return a temporarily deferred condition.

V\_SENDER\_RDSSC\_GREETINGS\_FAIL is set to GRAYLIST delay will be incremented!

REMOTE\_SMTP\_SENDER\_PERMANENT\_FAIL\_GRAYLIST\_DELAY if the SMTP transaction return a permanent faillure condition, except in RCPT TO fase. V\_SENDER\_RDSSC\_PERMANENT\_FAIL is set to GRAYLIST delay will be incremented!

REMOTE\_SMTP\_SENDER\_TRANSIENT\_FAIL\_GRAYLIST\_DELAY if the SMTP transaction return a transient faillure condition. V\_SENDER\_RDSSC\_TRANSIENT\_FAIL is set to GRAYLIST delay will be incremented!

REMOTE\_SMTP\_SENDER\_SKIP if this test was skipped due it was disabled by user/admin.

REMOTE\_SMTP\_SENDER\_SENDER\_AUTH if remote host passed in previous authentication check like SPF.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_ADMIN\_REJECT\_IP\_IN\_ADDRESS if remote host used literal IP in MAIL FROM address but the system administrator reject the use of literal IPs in mail addresses.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_NOT\_IN\_LITERAL\_FORMAT if remote host didn't use IP numbers in literal format in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_FORMAT\_INVALID if remote host used IP numbers with invalid format in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_INVALID\_IPV4 if remote host used an invalid IPv4 number in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_INVALID\_IPV4\_RFC1918 if remote host used and private RFC1918 IP number in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_INVALID\_IPV6 if remote host used an invalid IPv6 number in MAIL FROM address.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_IPV6\_ADDR\_IN\_IPV4\_CONNECTION if remote host used an IPv6 number in MAIL FROM address, but the current connection type is IPv4.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_IPV4\_ADDR\_IN\_IPV6\_CONNECTION if remote host used an IPv4 number in MAIL FROM address, but the current connection type is IPv6.

REMOTE\_SMTP\_SENDER\_CHECK\_IP\_NOT\_RESOLVE\_TO\_FQDN if remote host used an IP number, in MAIL FROM address, that don't resolve to a FQDN.

REMOTE\_SMTP\_SENDER\_ERROR on errors and a\_err\_war->antispam\_error is set!

#### Note:

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

#### 4.7.4.9 dns\_error\_t Sz MTA\_Heuristics\_Check\_Reverse\_DNS (char \* RFC5321\_Remote\_Hostname\_IP\_Information, control\_user\_search\_r\_t user\_opt, a\_ew\_status\_t \* a\_err\_war)

Check Reverse DNS in agreement with RFC-1912.

#### Parameters:

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

DNS\_CHECK\_OK if remote client is in accordance with RFC-1912.

DNS\_CHECK\_FAIL if remote client IP hasn't a reverse DNS pointer or if RFC5321\_Remote\_Hostname\_IP\_Information is null.

DNS\_CHECK\_TEMP\_FAIL on DNS temporary fails or on internal errors.

DNS\_CHECK\_FORGED if remote client has a reverse DNS pointer but it don't point to the IP number of the remote client or if it don't point to anything.

DNS\_CHECK\_DISABLED if Reverse DNS check is disabled.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This function is controled by user options!

**See also:**

Sz\_MTA\_Heuristics\_Search\_In\_Controlled\_Users\_List()(p. 80)

#### 4.7.4.10 void Sz\_MTA\_Heuristics\_Clear (a\_ew\_status\_t \* a\_err\_war)

Dealloc memory for rejected words and accepted hosts/users lists and for others init configs.

**Parameters:**

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

None.

**Note:**

This function can change a\_err\_war->antispam\_error and a\_err\_war->antispam\_warning, but not reset them!

**4.7.4.11** `graylist_return_t Sz_MTA_Heuristics_GrayList (char * RFC5321_Remote_Hostname_IP_Information, char * Mail_From, char * Rcpt_User, control_user_search_r_t user_opt, a_ew_status_t * a_err_war)`

Use Advanced GrayList support.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*Mail\_From* A char string pointer to MAIL FROM user information.

*Rcpt\_User* A char string pointer to RCPT TO user information.

*user\_opt* A `control_user_search_r_t` pointer to rcpt user options.

*a\_err\_war* A `a_ew_status_t` pointer to error status.

**Returns:**

GRAYLIST\_IN if IP/SENDER/RCPT was added to GrayList or found and delay time wasn't reached.

GRAYLIST\_OUT if IP/SENDER/RCPT was found and removed from GrayList because delay time was reached if GrayList support is disabled, return A\_FALSE always on errors.

GRAYLIST\_SENDER\_AUTH if SENDER was authenticated previously by any authentication method (like SPF).

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD have one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

*Mail\_From* SHOULD have the format: `user@anything.foo.faa` or `<user@anything.foo.faa>`.

*Rcpt\_User* SHOULD be anything like: `anybody@somewhere.foo.net` or `<anybody@somewhere.foo.net>`.

This function is controlled by user options!

**See also:**

`Sz_MTA_Heuristics_Search_In_Controlled_Users_List()`(p. 80)

#### 4.7.4.12 `a_boolean_t Sz_MTA_Heuristics_GrayList_Clean (a_ew_status_t * a_err_war)`

Clean up the GrayList cache directory and its entries.

##### Parameters:

`a_err_war` A `a_ew_status_t` pointer to error status.

##### Returns:

A\_TRUE if clean up with success.

A\_FALSE if not clean up performed.

#### 4.7.4.13 `v_helo_r_t Sz_MTA_Heuristics_Helo_Check (char * RFC5321_Remote_Hostname_IP_Information, char * Helo, control_user_search_r_t user_opt, a_ew_status_t * a_err_war)`

Verify HELO SMTP command for bad things.

##### Parameters:

`RFC5321_Remote_Hostname_IP_Information` A char string pointer to reverse DNS information.

`Helo` A char string pointer to HELO information.

`user_opt` A `control_user_search_r_t` pointer to rcpt user options.

`a_err_war` A `a_ew_status_t` pointer to error status.

##### Returns:

HELO\_OK if helo argument is verified with success.

HELO\_ERROR on errors and `a_err_war->antispam_error` is set!

HELO\_IP\_NOT\_LITERAL if IP numbers passed is not in literal format.

HELO\_IP\_FORMAT\_INVALID if IP numbers passed has invalid format.

HELO\_IPV4\_RFC1918 if IP IPv4 IP number is invalid.

HELO\_DOMAIN\_HAS\_PROHIBITED\_CHARS if FQDN has prohibited characteres, in case of a prohibited character, the `*prohibited_char` pointer is set.

HELO\_DOMAIN\_HAS\_NOT\_DOT if FQDN has not dot points.

HELO\_DOMAIN\_START\_WITH\_DOT if FQDN has dot point at the begin.

HELO\_DOMAIN\_END\_WITH\_DOT if FQDN has dot point at the end.

HELO\_DOMAIN\_RESOLVE\_INVALID\_IPV4 if FQDN points to an invalid IPv4 number.

HELO\_DOMAIN\_RESOLVE\_INVALID\_IPV6 if FQDN points to an invalid IPv6 number.

HELO\_DOMAIN\_NOT\_RESOLVE\_AND\_NOT\_HAS\_MX if FQDN don't resolv and MX record don't resolve.

HELO\_DOMAIN\_RESOLVE\_MX\_INVALID\_IPV4 if MX resolve to an invalid IPv4 number.

HELO\_DOMAIN\_RESOLVE\_MX\_INVALID\_IPV6 if MX resolve to an invalid IPv6 number.

HELO\_HAS\_PROHIBITED\_IP\_NETWORK if literal IP number match with bad helo list, in this case the denied IP/Network is pointed by `*bad_helo_domain_ptr` pointer.

HELO\_PROHIBITED\_ALL\_IPV4\_NUMBERS if all literal IPv4 numbers are prohibited in helo command.

HELO\_PROHIBITED\_ALL\_IPV6\_NUMBERS if all literal IPv6 numbers are prohibited in helo command.

HELO\_HAS\_PROHIBITED\_DOMAIN if FQDN match with bad helo list, in this case of the denied domain is pointed by `*bad_helo_domain_ptr` pointer.

**Note:**

RFC5321\_Remote\_Hostname\_IP\_Information SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

Helo SHOULD be not null string.

This function is not in agreement with RFC-1123!!!

**See also:**

`Sz_MTA_Heuristics_Search_In_Controlled_Users_List()`(p. 80)

**4.7.4.14 a\_boolean\_t Sz\_MTA\_Heuristics\_Init (a\_ew\_status\_t \* a\_err\_war)**

Init configs, rejected words and accepted hosts/users lists.

**Parameters:**

*a\_err\_war* A *a\_ew\_status\_t* pointer to error status.

**Returns:**

A\_TRUE if all tasks were performed with success.

A\_FALSE if get errors, like out of memory.

**4.7.4.15 a\_boolean\_t Sz\_MTA\_Heuristics\_Rebuild (a\_ew\_status\_t \* a\_err\_war)**

Rebuild system configurations or control\_user list and private users configurations.

**Parameters:**

*a\_err\_war* A *a\_ew\_status\_t* pointer to error status.

**Returns:**

A\_TRUE if all tasks were performed with success.

A\_FALSE if get errors, like out of memory.

**4.7.4.16** `a_boolean_t Sz_MTA_Heuristics_Search_In_Accepted_Hosts_List`  
 (`char * RFC5321_Remote_Hostname_IP_Information`, `a_ew_status_t`  
 \* `a_err_war`)

Find host in accepted hosts lists.

**Parameters:**

`RFC5321_Remote_Hostname_IP_Information` A char string pointer to reverse DNS information.

`a_err_war` A `a_ew_status_t` pointer to error status.

**Returns:**

A\_TRUE if found a host in accepted hosts list.

A\_FALSE if not, on errors or if `RFC5321_Remote_Hostname_IP_Information` is null.

**Note:**

`RFC5321_Remote_Hostname_IP_Information` SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This function is controled by user options!

**See also:**

`Sz_MTA_Heuristics_Search_In_Controlled_Users_List()`(p. 80)

**4.7.4.17** `control_user_search_r_t Sz_MTA_Heuristics_Search_In_Controlled_Users_List`  
 (`char * RFC5321_Remote_Hostname_IP_Information`, `char`  
 \* `Rcpt_User`, `a_ew_status_t` \* `a_err_war`)

Find user in controled users lists. Return values below are combinated and should be verified using AND (&) operation.

**Parameters:**

***RFC5321\_Remote\_Hostname\_IP\_Information*** A char string pointer to reverse DNS information.

***Rcpt\_User*** A char string pointer to user information.

***a\_err\_war*** A `a_ew_status_t` pointer to error status.

**Returns:**

**V\_NO\_USER** if no found user in controled users list.

**V\_NONE** if found user and no check will be procced.

**V\_INVALID** if found user and he is an invalid user.

**V\_RESTRICT** if found user and he is a restricted user.

**V\_RESTRICT\_POST** if found user and he has restricted post rights (not implemented yet).

**V\_WARNING\_INSTEAD\_REJECT** if found user and he don't want to reject messages, but add a warning in the subject of message (not implemented yet).

**V\_REVERSE** if found user and check reverse.

**V\_PROHIBITED\_FQDN\_STRINGS** if found user and he want to check if remote host has a prohibited FQDN string in reverse DNS (typically dialup strings), if reverse DNS exist.

**V\_BLACKLIST\_NO** if found user and NO use blacklist.

**V\_BLACKLIST\_YES** if found user and USE blacklist.

**V\_BLACKLIST\_PERSIST** if found user and USE blacklist in PERSIST mode.

**V\_BLACKLIST\_ADMIN\_DECIDES** if found user and USE blacklist in ADMIN\_DECIDES mode.

**V\_GRAYLIST** if found user and use graylist.

**V\_WHITELIST\_SPF\_NO** if found user and he don't want to use whitelist (based on SPF - Sender Policy Framework) (not implemented yet, only for LibAntispam 1.3.0 or up).

**V\_WHITELIST\_SPF\_YES** if found user and he want to use whitelist (not implemented yet, only for LibAntispam 1.3.0 or up).

**V\_WHITELIST\_SPF\_PERSIST** if found user and he want to USE whitelist in PERSISTENT mode (refuse mail while whitelist return TRY\_AGAIN (temporary DNS error)) (not implemented yet, only for LibAntispam 1.3.0 or up).

**V\_WHITELIST\_SPF\_ADMIN\_DECIDES** if found user and he want to USE whitelist in ADMIN\_DECIDES mode (lets system administrator choose if use whitelist with/without PERSISTENT mode or not use whitelist) (not implemented yet, only for LibAntispam 1.3.0 or up).

**V\_BAD\_HELO** if found user and he want to restrict bad HELO command use.

**V\_REMOTE\_DOMAIN\_SMTP\_SENDER\_CHECKING** if found user and he want to verify if remote domain or its MX accept SMTP connections for sender address passed in MAIL FROM cmd. (not implemented yet, only for a future version of LibAntispam).

**V\_FROM** if found user and he want to verify mail from consistency.

**V\_USER\_RESTRICT\_NULL\_REVERSE\_PATH\_USE** if user want to restrict null-reverse path.

**V\_USER\_RESTRICT\_SENDER\_USING\_SPF** if user want to use SPF to restrict bad senders.

**V\_USER\_RESTRICT\_LOCAL\_DOMAIN\_USE** if user want to restrict local domain use.

**V\_USER\_RESTRICT\_PROHIBITED\_CHARS\_USE** if user want to restrict prohibited chars.

**V\_USER\_RESTRICT\_BAD\_MAIL\_FROM\_USE** if user want to restrict bad MAIL FROM cmd.

**V\_HEADER\_ANALYZE** if found user and use realtime header analyze (not implemented yet!)

**V\_ERROR** on errors (may be synonym to V\_NO\_USER. If list is empty, `antispam_warning` is set).

**Note:**

global `_user_search_return_code` is set with return combined values above.  
`Rcpt_User` SHOULD be an E-mail address, like: `anybody@somewhere.foo` or `<anybody@somewhere.foo>`.

**4.7.4.18** `char * Sz_MTA_Heuristics_Search_In_Rejected_List`  
 (`char * RFC5321_Remote_Hostname_IP_Information`,  
`control_user_search_r_t user_opt`, `a_ew_status_t * a_err_war`)

Find a domain's substring in rejected words list.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*user\_opt* A `control_user_search_r_t` pointer to rcpt user options.

*a\_err\_war* A `a_ew_status_t` pointer to error status.

**Returns:**

NULL if no domain's substring match with rejected words or on errors.

NULL\_DNS\_INFORMATION if `RFC5321_Remote_Hostname_IP_Information` is null.

SPECIAL\_COMMAND\_RETURN if domain match in special command search and `antispam_warning` can be set with two values: `SU_CNS_FOUND_NUMERIC_SETS` if an wanted numeric sets were found or `SU_CNG_FOUND_NUMERIC_GROUP` if an wanted numeric group was found.

The prohibited word, if it was found in rejected words list.

**Note:**

`RFC5321_Remote_Hostname_IP_Information` SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

This function is controled by user options!

**See also:**

`Sz_MTA_Heuristics_Search_In_Controlled_Users_List()`(p. 80)

**4.7.4.19** `a_boolean_t Sz_MTA_Heuristics_Sendmail (char * RFC5321_Remote_Hostname_IP_Information, char * Mail_From, char * Rcpt_User, char * message, int message_type, control_user_search_r_t user_opt, a_ew_status_t * a_err_war)`

Send a warning mail to somebody.

**Parameters:**

*RFC5321\_Remote\_Hostname\_IP\_Information* A char string pointer to reverse DNS information.

*Mail\_From* A char string pointer to MAIL FROM user information.

*Rcpt\_User* A char string pointer to RCPT TO user information.

*message* A char string pointer to warning string message.

*message\_type* A int number to message type.

*user\_opt* A control\_user\_search\_r\_t pointer to rcpt user options.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

A\_TRUE if mail message was sent with success.

A\_FALSE if not or on errors.

**Note:**

*RFC5321\_Remote\_Hostname\_IP\_Information* SHOULD has one of the formats below:

IPv4: [xxx.xxx.xxx.xxx]

IPv4: anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv4: IDENT:anybody@anything.foo.org [xxx.xxx.xxx.xxx]

IPv6: [IPv6:XXXX:XXXX:XXXX::]

IPv6: anything.foo.org [IPv6:XXXX:XXXX:XXXX::]

IPv6: anybody@anything.foo.org [IPv6:XXXX:XXXX::]

IPv6: IDENT:anybody@anything.foo.org [IPv6:XXXX:XXXX:XXXX:XXXX::]

**Note:**

*Mail\_From* SHOULD has the format: *user@anything.faa* or *<user@anything.faa>*.

*Rcpt\_User* SHOULD be anything like: *anybody@somewhere.foo* or *<anybody@somewhere.foo>*.

This function is controled by user options!

**See also:**

*Sz\_MTA\_Heuristics\_Search\_In\_Controlled\_Users\_List()*(p.80)

**4.7.4.20** `a_boolean_t Sz_MTA_Heuristics_Vrfy_For_Recipient_User_Deny_SMTP_DATA (char * Rcpt_To, a_ew_status_t * a_err_war)`

Verify if the user part name of recipient information match with a user, in internal list, that is prohibited to do the DATA SMTP command (should be implemented in MTA, but is quit simple, we need only return a 451 that is the temporary error code in SMTP messages).

**Parameters:**

*Rcpt\_To* A char string pointer to address passed in RCPT TO command.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

A\_TRUE if match with an user in internal list.

A\_FALSE if not or on errors.

**Note:**

This is a simple function independent from LibAntispam admin or users options, so we don't need to verify things like race or rebuild conditions.

This function is a front-end to Sz\_MTA\_Heuristics\_Vrfy\_For\_User\_Deny\_SMTP\_DATA()

**4.7.4.21** `a_boolean_t Sz_MTA_Heuristics_Vrfy_For_Sender_User_Deny_SMTP_DATA (char * Mail_From, a_ew_status_t * a_err_war)`

Verify if the user part name of sender information match with a user, in internal list, that is prohibited to do the DATA SMTP command (should be implemented in MTA, but is quit simple, we need only return a 550 that is the permanent error code in SMTP messages when DATA command is used).

**Parameters:**

*Mail\_From* A char string pointer to address passed in MAIL FROM command.

*a\_err\_war* A a\_ew\_status\_t pointer to error status.

**Returns:**

A\_TRUE if match with an user in internal list.

A\_FALSE if not or on errors.

**Note:**

This is a simple function independent from LibAntispam admin or users options, so we don't need to verify things like race or rebuild conditions.

This function is a front-end to Sz\_MTA\_Heuristics\_Vrfy\_For\_User\_Deny\_SMTP\_DATA()

# Index

- ~SzAntispamMTA
  - SzAntispamMTA, 21
- a\_antispam\_error
  - SzAntispam, 12
- a\_antispam\_warning
  - SzAntispam, 12
- a\_ew\_status\_s, 5
  - antispam\_error, 5
  - antispam\_warning, 5
- a\_mh\_antispam\_error
  - SzAntispamMTA, 21
- a\_mh\_antispam\_warning
  - SzAntispamMTA, 21
- a\_mh\_str\_antispam\_error
  - SzAntispamMTA, 21
- a\_mh\_str\_antispam\_warning
  - SzAntispamMTA, 21
- a\_str\_antispam\_error
  - SzAntispam, 12
- a\_str\_antispam\_warning
  - SzAntispam, 12
- antispam++, 39
- antispam++.h, 40
- antispam.h, 41
  - multi\_rsbl\_search\_t, 42
  - rsbl\_search\_t, 43
  - Sz\_Check\_FQDN\_or\_IP\_in\_-  
Multiples\_RSBL, 43
  - Sz\_Check\_FQDN\_or\_IP\_in\_RSBL,  
44
  - Sz\_Check\_FQDN\_or\_IPv6\_in\_-  
Multiples\_RSBL, 44
  - Sz\_Check\_FQDN\_or\_IPv6\_in\_-  
RSBL, 45
  - Sz\_Check\_Valid\_FQDN, 46
  - Sz\_Check\_Valid\_FQDN\_or\_IP, 47
  - Sz\_Check\_Valid\_IP\_Number, 47
  - Sz\_Check\_Valid\_IPv6\_Number, 48
- antispam\_error
  - a\_ew\_status\_s, 5
- antispam\_error.h, 49
  - str\_antispam\_error, 54
  - str\_antispam\_warning, 54
- antispam\_warning
  - a\_ew\_status\_s, 5
- bad\_helo\_domain\_ptr
  - v\_helo\_r\_s, 37
- bad\_mail\_from\_domain
  - v\_mail\_from\_s, 38
- Check\_FQDN\_or\_IP\_in\_Multiples\_-  
RSBL
  - SzAntispam, 12
- Check\_FQDN\_or\_IP\_in\_RSBL
  - SzAntispam, 13
- Check\_FQDN\_or\_IPv6\_in\_Multiples\_-  
RSBL
  - SzAntispam, 14
- Check\_FQDN\_or\_IPv6\_in\_RSBL
  - SzAntispam, 15
- Check\_Valid\_FQDN
  - SzAntispam, 16
- Check\_Valid\_FQDN\_or\_IP
  - SzAntispam, 16
- Check\_Valid\_IP\_Number
  - SzAntispam, 17
- Check\_Valid\_IPv6\_Number
  - SzAntispam, 17
- control\_user\_search\_r\_s, 6
  - global\_user\_found\_in\_reduced\_-  
format, 6
  - global\_user\_search\_position, 6
  - global\_user\_search\_return\_code, 6
  - global\_user\_search\_type, 6
- control\_user\_search\_r\_t
  - mta-heuristics.h, 67
- deny\_ip\_network\_ptr
  - prohibited\_ip\_net\_r\_s, 9
- dns\_fail
  - multi\_rsbl\_search\_s, 8
- global\_user\_found\_in\_reduced\_format
  - control\_user\_search\_r\_s, 6
- global\_user\_search\_position
  - control\_user\_search\_r\_s, 6
- global\_user\_search\_return\_code
  - control\_user\_search\_r\_s, 6
- global\_user\_search\_type

- control\_user\_search\_r\_s, 6
- has\_or\_belongs\_prohibited\_ip\_or\_network
  - prohibited\_ip\_net\_r\_s, 9
- Heuristics\_BlackList
  - SzAntispamMTA, 22
- Heuristics\_Check\_For\_My\_or\_Friend\_IP\_Network
  - SzAntispamMTA, 22
- Heuristics\_Check\_For\_Prohibited\_IP\_Network
  - SzAntispamMTA, 23
- Heuristics\_Check\_Good\_Mail\_From
  - SzAntispamMTA, 24
- Heuristics\_Check\_Local\_User\_Status\_In\_Mail\_From
  - SzAntispamMTA, 25
- Heuristics\_Check\_Local\_User\_Status\_In\_Rcpt\_To
  - SzAntispamMTA, 25
- Heuristics\_Check\_Mail\_From
  - SzAntispamMTA, 26
- Heuristics\_Check\_Remote\_SMTP\_Sender
  - SzAntispamMTA, 27
- Heuristics\_Check\_Reverse\_DNS
  - SzAntispamMTA, 28
- Heuristics\_Clear
  - SzAntispamMTA, 29
- Heuristics\_GrayList
  - SzAntispamMTA, 29
- Heuristics\_GrayList\_Clean
  - SzAntispamMTA, 30
- Heuristics\_Helo\_Check
  - SzAntispamMTA, 30
- Heuristics\_Init
  - SzAntispamMTA, 31
- Heuristics\_Rebuild
  - SzAntispamMTA, 32
- Heuristics\_Search\_In\_Accepted\_Hosts\_List
  - SzAntispamMTA, 32
- Heuristics\_Search\_In\_Controlled\_Users\_List
  - SzAntispamMTA, 32
- Heuristics\_Search\_In\_Rejected\_List
  - SzAntispamMTA, 34
- Heuristics\_Sendmail
  - SzAntispamMTA, 34
- Heuristics\_Vrfy\_For\_Recipient\_User\_Deny\_SMTP\_DATA
  - SzAntispamMTA, 35
- Heuristics\_Vrfy\_For\_Sender\_User\_Deny\_SMTP\_DATA
  - SzAntispamMTA, 35
- SzAntispamMTA, 36
- ip\_rsbl\_address\_ptr
  - multi\_rsbl\_search\_s, 8
  - rsbl\_search\_s, 10
- message
  - multi\_dnsbl\_s, 7
- mta-heuristics++, 55
- mta-heuristics++.h, 56
- mta-heuristics.h, 57
  - control\_user\_search\_r\_t, 67
  - multi\_dnsbl\_t, 67
  - prohibited\_ip\_net\_r\_t, 67
  - Sz\_MTA\_Heuristics\_BlackList, 68
  - Sz\_MTA\_Heuristics\_Check\_For\_My\_or\_Friend\_IP\_Network, 69
  - Sz\_MTA\_Heuristics\_Check\_For\_Prohibited\_IP\_Network, 70
  - Sz\_MTA\_Heuristics\_Check\_Good\_Mail\_From, 71
  - Sz\_MTA\_Heuristics\_Check\_Local\_User\_Status\_In\_Mail\_From, 72
  - Sz\_MTA\_Heuristics\_Check\_Local\_User\_Status\_In\_Rcpt\_To, 72
  - Sz\_MTA\_Heuristics\_Check\_Mail\_From, 73
  - Sz\_MTA\_Heuristics\_Check\_Remote\_SMTP\_Sender, 74
  - Sz\_MTA\_Heuristics\_Check\_Reverse\_DNS, 75
  - Sz\_MTA\_Heuristics\_Clear, 76
  - Sz\_MTA\_Heuristics\_GrayList, 76
  - Sz\_MTA\_Heuristics\_GrayList\_Clean, 77
  - Sz\_MTA\_Heuristics\_Helo\_Check, 78
  - Sz\_MTA\_Heuristics\_Init, 79
  - Sz\_MTA\_Heuristics\_Rebuild, 79
  - Sz\_MTA\_Heuristics\_Search\_In\_Accepted\_Hosts\_List, 79
  - Sz\_MTA\_Heuristics\_Search\_In\_Controlled\_Users\_List, 80
  - Sz\_MTA\_Heuristics\_Search\_In\_Rejected\_List, 82
  - Sz\_MTA\_Heuristics\_Sendmail, 82
  - Sz\_MTA\_Heuristics\_Vrfy\_For\_Recipient\_User\_Deny\_SMTP\_DATA, 83
  - Sz\_MTA\_Heuristics\_Vrfy\_For\_Sender\_User\_Deny\_SMTP\_DATA, 84
  - v\_helo\_r\_t, 67

- v\_mail\_from\_return\_t, 68
- v\_mail\_from\_t, 68
- mta\_log\_message
  - multi\_dnsbl\_s, 7
- multi\_dnsbl\_s, 7
  - message, 7
  - mta\_log\_message, 7
  - result, 7
- multi\_dnsbl\_t
  - mta-heuristics.h, 67
- multi\_rsbl\_search\_s, 8
  - dns\_fail, 8
  - ip\_rsbl\_address\_ptr, 8
  - result, 8
  - txt\_message\_ptr, 8
- multi\_rsbl\_search\_t
  - antispam.h, 42
- prohibited\_char
  - v\_helo\_r\_s, 37
  - v\_mail\_from\_s, 38
- prohibited\_ip\_net\_r\_s, 9
  - deny\_ip\_network\_ptr, 9
  - has\_or\_belongs\_prohibited\_ip\_or\_network, 9
- prohibited\_ip\_net\_r\_t
  - mta-heuristics.h, 67
- result
  - multi\_dnsbl\_s, 7
  - multi\_rsbl\_search\_s, 8
  - rsbl\_search\_s, 10
- return\_value
  - v\_helo\_r\_s, 37
  - v\_mail\_from\_s, 38
- rsbl\_search\_s, 10
  - ip\_rsbl\_address\_ptr, 10
  - result, 10
  - txt\_message\_ptr, 10
- rsbl\_search\_t
  - antispam.h, 43
- spf\_exp
  - v\_mail\_from\_s, 38
- str\_antispam\_error
  - antispam\_error.h, 54
- str\_antispam\_warning
  - antispam\_error.h, 54
- Sz\_Check\_FQDN\_or\_IP\_in\_Multiples\_RSBL
  - antispam.h, 43
- Sz\_Check\_FQDN\_or\_IP\_in\_RSBL
  - antispam.h, 44
- Sz\_Check\_FQDN\_or\_IPv6\_in\_Multiples\_RSBL
  - antispam.h, 44
- Sz\_Check\_FQDN\_or\_IPv6\_in\_RSBL
  - antispam.h, 45
- Sz\_Check\_Valid\_FQDN
  - antispam.h, 46
- Sz\_Check\_Valid\_FQDN\_or\_IP
  - antispam.h, 47
- Sz\_Check\_Valid\_IP\_Number
  - antispam.h, 47
- Sz\_Check\_Valid\_IPv6\_Number
  - antispam.h, 48
- Sz\_MTA\_Heuristics\_BlackList
  - mta-heuristics.h, 68
- Sz\_MTA\_Heuristics\_Check\_For\_My\_or\_Friend\_IP\_Network
  - mta-heuristics.h, 69
- Sz\_MTA\_Heuristics\_Check\_For\_Prohibited\_IP\_Network
  - mta-heuristics.h, 70
- Sz\_MTA\_Heuristics\_Check\_Good\_Mail\_From
  - mta-heuristics.h, 71
- Sz\_MTA\_Heuristics\_Check\_Local\_User\_Status\_In\_Mail\_From
  - mta-heuristics.h, 72
- Sz\_MTA\_Heuristics\_Check\_Local\_User\_Status\_In\_Rcpt\_To
  - mta-heuristics.h, 72
- Sz\_MTA\_Heuristics\_Check\_Mail\_From
  - mta-heuristics.h, 73
- Sz\_MTA\_Heuristics\_Check\_Remote\_SMTP\_Sender
  - mta-heuristics.h, 74
- Sz\_MTA\_Heuristics\_Check\_Reverse\_DNS
  - mta-heuristics.h, 75
- Sz\_MTA\_Heuristics\_Clear
  - mta-heuristics.h, 76
- Sz\_MTA\_Heuristics\_GrayList
  - mta-heuristics.h, 76
- Sz\_MTA\_Heuristics\_GrayList\_Clean
  - mta-heuristics.h, 77
- Sz\_MTA\_Heuristics\_Helo\_Check
  - mta-heuristics.h, 78
- Sz\_MTA\_Heuristics\_Init
  - mta-heuristics.h, 79
- Sz\_MTA\_Heuristics\_Rebuild
  - mta-heuristics.h, 79
- Sz\_MTA\_Heuristics\_Search\_In\_Accepted\_Hosts\_List
  - mta-heuristics.h, 79

- Sz\_MTA\_Heuristics\_Search\_In\_-
  - Controlled\_Users\_List
  - mta-heuristics.h, 80
- Sz\_MTA\_Heuristics\_Search\_In\_-
  - Rejected\_List
  - mta-heuristics.h, 82
- Sz\_MTA\_Heuristics\_Sendmail
  - mta-heuristics.h, 82
- Sz\_MTA\_Heuristics\_Vrfy\_For\_-
  - Recipient\_User\_Deny\_SMTP\_-
  - DATA
  - mta-heuristics.h, 83
- Sz\_MTA\_Heuristics\_Vrfy\_For\_Sender\_-
  - User\_Deny\_SMTP\_DATA
  - mta-heuristics.h, 84
- SzAntispam, 11
- SzAntispam
  - a\_antispam\_error, 12
  - a\_antispam\_warning, 12
  - a\_str\_antispam\_error, 12
  - a\_str\_antispam\_warning, 12
  - Check\_FQDN\_or\_IP\_in\_Multiples\_-
  - RSBL, 12
  - Check\_FQDN\_or\_IP\_in\_RSBL, 13
  - Check\_FQDN\_or\_IPv6\_in\_-
  - Multiples\_RSBL, 14
  - Check\_FQDN\_or\_IPv6\_in\_RSBL, 15
  - Check\_Valid\_FQDN, 16
  - Check\_Valid\_FQDN\_or\_IP, 16
  - Check\_Valid\_IP\_Number, 17
  - Check\_Valid\_IPv6\_Number, 17
- SzAntispamMTA, 19
  - SzAntispamMTA, 21
- SzAntispamMTA
  - ~SzAntispamMTA, 21
  - a\_mh\_antispam\_error, 21
  - a\_mh\_antispam\_warning, 21
  - a\_mh\_str\_antispam\_error, 21
  - a\_mh\_str\_antispam\_warning, 21
  - Heuristics\_BlackList, 22
  - Heuristics\_Check\_For\_My\_or\_-
  - Friend\_IP\_Network, 22
  - Heuristics\_Check\_For\_Prohibited\_-
  - IP\_Network, 23
  - Heuristics\_Check\_Good\_Mail\_From,
  - 24
  - Heuristics\_Check\_Local\_User\_-
  - Status\_In\_Mail\_From, 25
  - Heuristics\_Check\_Local\_User\_-
  - Status\_In\_Rcpt\_To, 25
  - Heuristics\_Check\_Mail\_From, 26
  - Heuristics\_Check\_Remote\_SMTP\_-
  - Sender, 27
  - Heuristics\_Check\_Reverse\_DNS, 28
  - Heuristics\_Clear, 29
  - Heuristics\_GrayList, 29
  - Heuristics\_GrayList\_Clean, 30
  - Heuristics\_Helo\_Check, 30
  - Heuristics\_Init, 31
  - Heuristics\_Rebuild, 32
  - Heuristics\_Search\_In\_Accepted\_-
  - Hosts\_List, 32
  - Heuristics\_Search\_In\_Controlled\_-
  - Users\_List, 32
  - Heuristics\_Search\_In\_Rejected\_List,
  - 34
  - Heuristics\_Sendmail, 34
  - Heuristics\_Vrfy\_For\_Recipient\_-
  - User\_Deny\_SMTP\_DATA, 35
  - Heuristics\_Vrfy\_For\_Sender\_User\_-
  - Deny\_SMTP\_DATA, 36
- SzAntispamMTA, 21
- txt\_message\_ptr
  - multi\_rsbl\_search\_s, 8
  - rsbl\_search\_s, 10
- unwanted\_char
  - v\_mail\_from\_s, 38
- v\_helo\_r\_s, 37
  - bad\_helo\_domain\_ptr, 37
  - prohibited\_char, 37
  - return\_value, 37
- v\_helo\_r\_t
  - mta-heuristics.h, 67
- v\_mail\_from\_return\_t
  - mta-heuristics.h, 68
- v\_mail\_from\_s, 38
  - bad\_mail\_from\_domain, 38
  - prohibited\_char, 38
  - return\_value, 38
  - spf\_exp, 38
  - unwanted\_char, 38
- v\_mail\_from\_t
  - mta-heuristics.h, 68