

Network Working Group  
Request for Comments: 4012  
Updates: 2725, 2622  
Category: Standards Track

L. Blunk  
Merit Network  
J. Damas  
Internet Systems Consortium  
F. Parent  
Hexago  
A. Robachevsky  
RIPE NCC  
March 2005

## Routing Policy Specification Language next generation (RPSLNg)

### Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

### Copyright Notice

Copyright (C) The Internet Society (2005).

### Abstract

This memo introduces a new set of simple extensions to the Routing Policy Specification Language (RPSL), enabling the language to document routing policies for the IPv6 and multicast address families currently used in the Internet.

### Table of Contents

1.	Introduction . . . . .	2
2.	Specifying routing policy for different address families . . . . .	2
2.1.	Ambiguity Resolution . . . . .	3
2.2.	The afi dictionary attribute . . . . .	3
2.3.	RPSL dictionary extensions . . . . .	4
2.4.	IPv6 RPSL types . . . . .	4
2.5.	mp-import, mp-export, and mp-default . . . . .	4
2.5.1.	<mp-peering> . . . . .	6
2.5.2.	<mp-filter> . . . . .	6
2.5.3.	Policy examples . . . . .	7
3.	route6 Class . . . . .	7
4.	Updates to existing Classes to support the extensions . . . . .	8
4.1.	as-set Class . . . . .	8
4.2.	route-set Class . . . . .	9

4.3.	filter-set Class . . . . .	9
4.4.	peering-set Class . . . . .	9
4.5.	inet-rtr Class . . . . .	10
4.6.	rtr-set Class . . . . .	11
5.	RFC 2725 Extensions . . . . .	11
5.1.	Authorization model for route6 Objects . . . . .	13
6.	Security Considerations . . . . .	13
7.	Acknowledgments . . . . .	14
8.	References . . . . .	14
8.1.	Normative References . . . . .	14
8.2.	Informative References . . . . .	14
	Authors' Addresses . . . . .	15
	Full Copyright Statement . . . . .	16

## 1. Introduction

RFC 2622 [1] defines the RPSL language for the IPv4 unicast routing protocols and provides a series of guidelines for extending the RPSL language itself. Additionally, security extensions to the RPSL language are specified in RFC 2725 [2].

This document proposes to extend RPSL according to the following goals and requirements:

- o Provide RPSL extensibility in the dimension of address families, specifically, to allow users to document routing policy for IPv6 and multicast.
- o Extensions should be backward compatible with minimal impact on existing tools and processes, following Section 10 of RFC 2622 [1] for guidelines on extending RPSL.
- o Maintain clarity and non-ambiguity: RPSL information is used by humans in addition to software tools.
- o Minimize duplication of information, particularly when routing policies for different address families are the same.

The addition of IPv6 and multicast support to RPSL leads to four distinct routing policies that need to be distinguished in this specification, namely, (IPv4 {unicast|multicast}, IPv6 {unicast|multicast}).

## 2. Specifying Routing Policy for Different Address Families

Routing policy is currently specified in the aut-num class using "import:", "export:", and "default:" attributes. Sometimes it is important to distinguish policy for different address families, as well as a unicast routing policy from a multicast one.

Although the syntax of the existing import, export, and default attributes could be extended, this would present backward compatibility issues and could undermine clarity in the expressions.

Keeping this in mind, the "import:", "export:", and "default:" attributes implicitly specify IPv4 unicast policy and will remain as previously defined in RPSL, and new multi-protocol (prefixed with the string "mp-") attributes will be introduced. These new "mp-" attributes are described below.

## 2.1. Ambiguity Resolution

The same peering can be covered by more than one multi-protocol policy attribute or by a combination of multi-protocol policy attributes (when specifying IPv4 unicast policy) and the previously defined IPv4 unicast policy attributes. In these cases, implementations should follow the specification-order rule as defined in Section 6.4 of RFC 2622 [1]. To break the ambiguity, the action corresponding to the first peering specification is used.

## 2.2. The afi Dictionary Attribute

This section introduces a new dictionary attribute:

Address Family Identifier, <afi>, is an RPSL list of address families for which a given routing policy expression should be evaluated. <afi> is optional within the new multi-protocol attributes introduced in the aut-num class. A pseudo identifier named "any" is defined to allow for more compact policy expressions with converged routing policy.

The possible values for <afi> are as follows:

```
ipv4.unicast
ipv4.multicast
ipv4 (equivalent to ipv4.unicast, ipv4.multicast)
ipv6.unicast
ipv6.multicast
ipv6 (equivalent to ipv6.unicast, ipv6.multicast)
any (equivalent to ipv4, ipv6)
any.unicast (equivalent to ipv4.unicast, ipv6.unicast)
any.multicast (equivalent to ipv4.multicast, ipv6.multicast)
```

Appearance of these values in an attribute must be preceded by the keyword afi.

An <afi-list> is defined as a comma-separated list of one or more afi values.

### 2.3. RPSL Dictionary Extensions

In order to support IPv6 addresses specified with the next-hop rp-attribute, a new predefined dictionary type entitled "ipv6\_address" is added to the RPSL dictionary. The definition of this type is taken from Section 2.2 of RFC 3513 [3].

The next-hop rp-attribute is expanded in the dictionary as follows:

```
rp-attribute: # next hop router in a static route
              next-hop
              operator=(union ipv4_address, ipv6_address, enum[self])
```

A new value has been added for the <protocol> dictionary specification:  
MPBGP

MPBGP is understood to be BGP4 with multi-protocol extensions (often referred to as BGP4+). BGP4+ could not be used, as the '+' character is not allowed by the RPSL specification in protocol names.

### 2.4. IPv6 RPSL Types

This document will reference three new IPv6 RPSL types, namely, <ipv6-address>, <ipv6-address-prefix>, and <ipv6-address-prefix-range>. The <ipv6-address> and <ipv6-address-prefix> types are defined in Sections 2.2 and 2.3 of RFC 3513 [3]. The <ipv6-address-prefix-range> type adds a range operator to the <ipv6-address-prefix> type. The range operator is defined in Section 2 of RFC 2622 [1].

### 2.5. mp-import, mp-export, and mp-default

Three new policy attributes are introduced in the aut-num Class:

```
mp-import:
mp-export:
mp-default:
```

These attributes incorporate the afi (address-family) specification. Note that the afi specification is optional. If no afi specification is present, the policy expression is presumed to apply to all protocol families, namely, ipv4.unicast, ipv4.multicast, ipv6.unicast, and ipv6.multicast. This is the equivalent of the afi specification "afi any". The mp-import and mp-export attributes have both a basic policy specification and a more powerful structured policy specification.

The syntax for the mp-default attribute and the basic policy specification of the mp-import and mp-export attributes is as follows:

Attribute	Value	Type
mp-import	[protocol <protocol-1>] [into <protocol-2>] [afi <afi-list>] from <mp-peering-1> [action <action-1>; ... <action-N>;] ... from <mp-peering-M> [action <action-1>; ... <action-N>;] accept <mp-filter> [;]	optional, multi-valued
mp-export	[protocol <protocol-1>] [into <protocol-2>] [afi <afi-list>] to <mp-peering-1> [action <action-1>; ... <action-N>;] ... to <mp-peering-M> [action <action-1>; ... <action-N>;] announce <mp-filter> [;]	optional, multi-valued
mp-default	[afi <afi-list>] to <mp-peering> [action <action-1>; ... <action-N>;] [networks <mp-filter>]	optional, multi-valued

The mp-import and mp-export policies can be structured. As with RFC 2622 [1], structured policies are recommended only to advanced RPSL users. The mp-import structured policy syntax is defined below. Please note the semicolon at the end of an <import-factor> is mandatory for structured policy expressions, while being optional on non-structured policy expressions. The mp-export structured policy syntax is expressed symmetrically to the mp-import attribute. The structured syntax allows exceptions and refinements to policies by use of the "except" and "refine" keywords. Further, the exceptions and refinements may specify an optional "afi" list to restrict the policy expression to particular address families.

Note that the definition allows subsequent or "cascading" refinements and exceptions. RFC 2622 [1] incorrectly refers to these as "nested" expressions. The syntax does not allow true nested expressions.

```
<import-factor> ::=
    from <mp-peering-1> [action <action-1>; ... <action-M>;]
    . . .
    from <mp-peering-N> [action <action-1>; ... <action-K>;]
    accept <mp-filter>;

<import-term> ::= import-factor |
    {
        <import-factor-1>
```

```

    . . .
    <import-factor-N>
    }

<import-expression> ::= <import-term> |
    <import-term> EXCEPT <afi-import-expression> |
    <import-term> REFINE <afi-import-expression>

<afi-import-expression> ::= [afi <afi-list>] <import-expression>

mp-import: [protocol <protocol-1>] [into <protocol-2>]
    <afi-import-expression>

```

#### 2.5.1. <mp-peering>

<mp-peering> indicates the AS (and the router if present) and is defined as follows:

```

<mp-peering> ::= <as-expression> [<mp-router-expression-1>]
    [at <mp-router-expression-2>] | <peering-set-name>

```

where <as-expression> is an expression over AS numbers and AS sets using operators AND, OR, and EXCEPT, and <mp-router-expression> is an expression over router ipv4-addresses or ipv6-addresses, inet-rtr names, and rtr-set names using operators AND, OR, and EXCEPT. The binary "EXCEPT" operator is the set subtraction operator and has the same precedence as the operator AND (it is semantically equivalent to "AND NOT" combination). That is, "(AS65001 OR AS65002) EXCEPT AS65002" equals "AS65001".

#### 2.5.2. <mp-filter>

The <mp-filter> policy filter expression is derived from the RPSL <filter> policy filter expression defined in section 5.4 of RFC 2622 [1]. <mp-filter> extends the <filter> expression to allow the specification of IPv6 prefixes and prefix ranges. In particular, an Address-Prefix Set expression in an <mp-filter> expression may include both IPv4 and IPv6 prefixes or prefix ranges. <mp-filter> is otherwise identical to the RPSL <filter> expression. Address-Prefix Sets are enclosed in braces, '{' and '}'. The policy filter matches the set of routes whose destination address-prefix is in the set. For example:

```

{ 192.0.2.0/24, 2001:0DB8::/32 }
{ 2001:0DB8:0100::/48^+, 2001:0DB8:0200::/48^64 }

```

### 2.5.3. Policy Examples

The address family may be specified in subsequent refine or except policy expressions and is valid only within the policy expression that contains it.

Therefore, in the example

```
aut-num:      AS65534
mp-import:    afi any.unicast from AS65001 accept as-foo;
              except afi any.unicast {
                from AS65002 accept AS65226;
              } except afi ipv6.unicast {
                from AS65003 accept {2001:0DB8::/32};
              }
```

the last "except" is evaluated only for the IPv6 unicast address family, while other import-expressions are evaluated for both the IPv6 and IPv4 unicast address families.

The evaluation of a policy expression is done by evaluating each of its components. Evaluation of peering-sets and filter-sets is constrained by the address family. Such constraints may result in a "NOT ANY" <mp-filter> or invalid <mp-peering> depending on implicit or explicit definitions of the address family in the set. Conflicts with explicit or implicit declarations are resolved at runtime during the evaluation of a policy expression. An RPSL evaluation implementation may wish to issue a warning in the case of a "NOT ANY" <mp-filter>. The following mp-import policy contains an example of an <mp-filter> that should be evaluated as "NOT ANY":

```
aut-num: AS65002
mp-import: afi ipv6.unicast from AS65001 accept {192.0.2.0/24}
```

### 3. route6 Class

The route6 class is the IPv6 equivalent of the route class. As with the route class, the class key for the route6 class is specified by the route6 and origin attribute pair. Other than the route6 attribute, the route6 class shares the same attribute names with the route class. Although the attribute names remain identical, the inject, components, exports-comps, holes, and mnt-routes attributes must specify IPv6 prefixes and addresses rather than IPv4 prefixes and addresses. This requirement is reflected by the specification of <ipv6-router-expression>, <ipv6-filter>, and <ipv6-address-prefix> below. <ipv6-address-prefix> has been previously defined. <ipv6-filter> is related to <mp-filter> as defined above in Section 2.5.2, with the exception that only <ipv6-address-prefix> types are

permitted. Similarly, <ipv6-router-expression> is related to <mp-router-expression> as defined above in Section 2.5.1 with the exception that only <ipv6-address> types are permitted.

Attribute	Value	Type
route6	<ipv6-address-prefix>	mandatory, class key, single-valued
origin	<as-number>	mandatory, class key, single-valued
member-of	list of <route-set-name>	optional, multi-valued
inject	[at <ipv6-router-expression>] ... [action <action>] [upon <condition>]	optional, multi-valued
components	[ATOMIC] [[<ipv6-filter>] [protocol <protocol> <ipv6-filter> ...]]	optional, single-valued
aggr-bndry	<as-expression>	optional, single-valued
aggr-mtd	inbound or outbound [<as-expression>]	optional, single-valued
export-comps	<ipv6-filter>	optional, single-valued
holes	list of <ipv6-address-prefix>	optional, multi-valued
mnt-lower	list of <mntner-name>	optional, multi-valued
mnt-routes	list of <mntner-name> [{list of <ipv6-address-prefix-range>} or ANY]	optional, multi-valued

Example:

```
route6: 2001:0DB8::/32
origin: AS65001
```

## 4. Updates to Existing Classes to Support the Extensions

### 4.1. as-set Class

The as-set class defines a set of Autonomous Systems (AS), specified either directly by listing them in the members attribute or indirectly by referring to another as-set or using the mbrs-by-ref facility. More importantly, "In a context that expects a route set (e.g., members attribute of the route-set class), [...] an as-set AS-X defines the set of routes that are originated by the ASes in AS-X", (section 5.3 of RFC 2622 [1]).

The as-set class is therefore used to collect a set of route prefixes, which may be restricted to a specific address family.

The existing as-set class does not need any modifications. The evaluation of the class must be filtered to obtain prefixes belonging to a particular address family using the traditional filtering mechanism in use in Internet Routing Registry (IRR) systems today.



## 4.2. route-set Class

This class is used to specify a set of route prefixes.

A new attribute "mp-members:" is defined for this class. This attribute allows the specification of IPv4 or IPv6 address-prefix-ranges.

Attribute	Value	Type
mp-members	list of (<ipv4-address-prefix-range> or <ipv6-address-prefix-range> or <route-set-name> or <route-set-name><range-operator>)	optional, multi-valued

Example:

```
route-set:  rs-foo
mp-members: rs-bar
mp-members: 2001:0DB8::/32 # v6 member
mp-members: 192.0.2.0/24   # v4 member
```

## 4.3. filter-set Class

The new "mp-filter:" attribute defines the set's policy filter. A policy filter is a logical expression that when applied to a set of routes returns a subset of these routes. The relevant parts of the updated filter-set class are shown below:

Attribute	Value	Type
filter-set	<object-name>	mandatory, single-valued, class key
filter	<filter>	optional, single-valued
mp-filter	<mp-filter>	optional, single-valued

Where <mp-filter> is defined above in Section 2.5.2. While the "filter:" and "mp-filter:" attributes are of type "optional", a filter-set must contain one of these two attributes. Implementations should reject instances where both attributes are defined in an object, as the interpretation of such a filter-set is undefined.

## 4.4. peering-set Class

The peering set class is updated with a "mp-peering:" attribute.

Attribute	Value	Type
peering-set	<object-name>	mandatory, single-valued, class key
peering	<peering>	optional, multi-valued
mp-peering	<mp-peering>	optional, multi-valued

Example:

```
peering-set:    prng-ebgp-peers
mp-peering:     AS65002 2001:0DB8::1 at 2001:0DB8::2
```

With <mp-peering> defined as above in Section 2.5.1. While the "peering:" and "mp-peering:" attributes are of type "optional", a peering-set must contain at least one of these two attributes.

#### 4.5. inet-rtr Class

Two new attributes are introduced to the inet-rtr class -- "interface:", which allows the definition of generic interfaces, including the information previously contained in the "ifaddr:" attribute, as well as support for tunnel definitions; and "mp-peer:", which includes and extends the functionality of the existing "peer:" attribute. The syntax definition for the "interface:" attribute follows:

Attribute	Value	Type
interface	<ipv4-address> or <ipv6-address> masklen <mask> [action <action>] [tunnel <remote-endpoint-address>,<encapsulation>]	optional, multi-valued

The syntax allows native IPv4 and IPv6 interface definitions, as well as the definition of tunnels as virtual interfaces. Without the optional tunnel definition, this attribute allows the same functionality as the "ifaddr:" attribute but extends it to allow IPv6 addresses.

If the interface is a tunnel, the syntax is as follows:

<remote-endpoint-address> indicates the IPv4 or IPv6 address of the remote endpoint of the tunnel. The address family must match that of the local endpoint. <encapsulation> denotes the encapsulation used in the tunnel and is one of {GRE,IPinIP} (note that the outer and inner IP protocol versions can be deduced from the interface context -- for example, IPv6-in-IPv4 encapsulation is just IPinIP). Routing policies for these routers should be described in the appropriate classes (e.g., aut-num).

The "mp-peer:" attribute is defined below. The difference between this attribute and the "peer:" attribute is the inclusion of support for IPv6 addresses.

Attribute	Value	Type
mp-peer	<protocol> <ipv4-address> <options> or <protocol> <ipv6-address> <options> or <protocol> <inet-rtr-name> <options> or <protocol> <rtr-set-name> <options> or <protocol> <peering-set-name> <options>	optional, multi-valued

where <protocol> is a protocol name, and <options> is a comma-separated list of peering options for <protocol>, as provided in the RPSL dictionary.

#### 4.6. rtr-set Class

The rtr-set class is extended with a new attribute, "mp-members:". This attribute extends the original "members:" attribute by allowing the specification of IPv6 addresses. It is defined as follows:

Attribute	Value	Type
mp-members	list of (<inet-rtr-name> or <rtr-set-name> or <ipv4-address> or <ipv6-address>)	optional, multi-valued

#### 5. RFC 2725 Extensions

RFC 2725 [2] introduces an authorization model to address the integrity of policy expressed in routing registries. Two new attributes were defined to support this authorization model: the "mnt-routes" and "mnt-lower" attributes.

In RPSLng, these attributes are extended to the route6 and inet6num (described below) classes. Further, the syntax of the existing mnt-routes attribute is modified to allow the optional specification of IPv6 prefix range lists when present in inet6num, route6, and aut-num class objects. This optional list of prefix ranges is a comma-separated list enclosed in curly braces. In the aut-num class, the IPv6 prefix ranges may be mixed with IPv4 prefix ranges. The keyword "ANY" may also be used instead of prefix ranges. In the case of inet6num and route6 objects, "ANY" refers to all more specifics of the prefix in the class key field. For the aut-num class, "ANY" literally means any prefix. The default when no additional set items are specified is "ANY". An abbreviated definition of the aut-num class with the updated syntax for the mnt-routes attribute is presented below.

Attribute	Value	Type
aut-num	<as-number>	mandatory, class key, single-valued
mnt-routes	list of <mntner-name> [ {list of (<ipv6-address-prefix-range> or <ipv4-address-prefix-range>)} or ANY]	optional, multi-valued

The following is an example of mnt-routes usage. This example authorizes MAINT-65001 to create route6 objects with an origin AS of 65002 for IPv6 address prefixes within the 2001:0DB8::/32^+ range, and route objects with origin AS 65002 for IPv4 prefixes within the 192.0.2.0/24^+ range.

aut-num: AS65002

mnt-routes: MAINT-AS65001 {2001:0DB8::/32^+, 192.0.2.0/24^+}

Note, that the inclusion of IPv6 prefix ranges within a mnt-routes attribute in an aut-num object may conflict with existing implementations of RPSL that support only IPv4 prefix ranges. However, given the perceived lack of implementation of this optional prefix range list, it was considered more acceptable to extend the existing definition of the mnt-routes attribute in the aut-num class rather than to create a new attribute type.

Attribute	Value	Type
inet6num	<ipv6-address-prefix>	mandatory, single-valued, class key
netname	<netname>	mandatory, single-valued
descr	<free-form>	mandatory, multi-valued
country	<country-code>	mandatory, multi-valued
admin-c	<nic-handle>	mandatory, multi-valued
tech-c	<nic-handle>	mandatory, multi-valued
remarks	<free-form>	optional, multi-valued
notify	<email-address>	optional, multi-valued
mnt-lower	list of <mntner-name>	optional, multi-valued
mnt-routes	list of <mntner-name> [ {list of <ipv6-address-prefix-range>} or ANY]	optional, multi-valued
mnt-by	list of <mntner-name>	mandatory, multi-valued
changed	<email-address> <date>	mandatory, multi-valued
source	<registry-name>	mandatory, single-valued

The <country-code> must be a valid two-letter ISO 3166 country code identifier. <netname> is a symbolic name for the specified IPv6 address space. It does not have a restriction on RPSL reserved prefixes. These definitions are taken from the RIPE Database Reference Manual [4].

### 5.1. Authorization Model for route6 Objects

Deletion and update of a route6 object is not different from other objects, as defined in RFC 2725 [2]. Creation rules of a route6 object is replicated here from the corresponding rules for route object in RFC 2725 [2] section 9.9.

When a route6 object is added, the submission must satisfy two authentication criteria. It must match the authentication specified in the aut-num object and that specified in either a route6 object or, if no applicable route6 object is found, an inet6num object.

An addition is submitted with an AS number and IPv6 prefix as its key. If the aut-num object does not exist on a route6 to add, then the addition is rejected. If the aut-num exists, then the submission is checked against the applicable maintainers. A search is then done for the prefix, looking first for an exact match and then, failing that, for the longest prefix match less specific than the prefix specified. If this search succeeds, it will return one or more route6 objects. The submission must match an applicable maintainer in at least one of these route6 objects for the addition to succeed. If the search for a route6 object fails, then a search is performed for an inet6num object that exactly matches the prefix, or for the most specific inet6num less specific than the route6 object submission.

Once the aut-num and either a list of route6 objects or an inet6num is found, the authorization is taken from these objects. The applicable maintainer object is any referenced by the mnt-routes attributes. If one or more mnt-routes attributes are present in an object, the mnt-by or mnt-lower attributes are not considered. In the absence of a mnt-routes attribute in a given object, the first mnt-lower attributes are used (only if the given object is an inet6num object and it is less specific than the route6 object to be added). If no applicable mnt-lower attribute is found, then the mnt-by attributes are used for that object. The authentication must match one of the authorizations in each of the two objects.

## 6. Security Considerations

This document describes extensions to RFC 2622 [1] and RFC 2725 [2]. The extensions address the limitations of the aforementioned documents with respect to IPv6 and multicast. The extensions do not introduce any new security functionality or threats.

Although the extensions introduce no additional security threats, it should be noted that the original RFC 2622 [1] RPSL standard included several weak and/or vulnerable authentication mechanisms: first, the "MAIL-FROM" scheme, which can be easily defeated via source email address spoofing; second, the "CRYPT-PW" scheme, which is subject to dictionary attacks and password sniffing if RPSL objects are submitted via unencrypted channels such as email; and, finally, the "NONE" mechanism, which offers no protection for objects.

## 7. Acknowledgements

The authors wish to thank all the people who have contributed to this document through numerous discussions, particularly Ekaterina Petrusha, for highly valuable discussions and suggestions: Shane Kerr, Engin Gunduz, Marc Blanchet, and David Kessens who participated constructively in many discussions and Cengiz Alaettinoglu, who is still the reference in all things RPSL.

## 8. References

### 8.1. Normative References

- [1] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and M. Terpstra, "Routing Policy Specification Language (RPSL)", RFC 2622, June 1999.
- [2] Villamizar, C., Alaettinoglu, C., Meyer, D., and S. Murphy, "Routing Policy System Security", RFC 2725, December 1999.
- [3] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

### 8.2. Informative References

- [4] Damas, J. and A. Robachevsky, "RIPE Database Reference Manual", August 2002.

## Authors' Addresses

Larry Blunk  
Merit Network

EMail: [ljb@merit.edu](mailto:ljb@merit.edu)

Joao Damas  
Internet Systems Consortium

EMail: [Joao\\_Damas@isc.org](mailto:Joao_Damas@isc.org)

Florent Parent  
Hexago

EMail: [Florent.Parent@hexago.com](mailto:Florent.Parent@hexago.com)

Andrei Robachevsky  
RIPE NCC

EMail: [andrei@ripe.net](mailto:andrei@ripe.net)

## Full Copyright Statement

Copyright (C) The Internet Society (2005).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.



