RIPE Database Update Reference Manual

Abstract

This document describes how to update the latest version of the RIPE Database. This series uses the Routing Policy Specification Language (RPSL) [1] to represent many of the database objects. It uses the Routing Policy System Security (RPSS)[2] for authorisation. This provides better security for Internet Routing Registries (IRRs). It makes use of RPSL next generation specifications [14] allowing for registering of multicast and IPv6 routing policies. Though this document is self-contained, you may also read the RPSL [1] and RPSS [2] specifications. For a tutorial on RPSL, you can read the RPSL applications document [3].

Intended Audience

This reference manual is for intermediate and advanced users who update the RIPE Database. If you are new to the database, you might find the "RIPE Database User Manual: Getting Started" [5] to be a more helpful place to start.

Conventions Used in This Document

We use <label> for a placeholder or to indicate syntax.
We use [option] to indicate an optional text or command argument.
In object templates, we use square brackets "[ ]" to indicate an attribute type.

"RIPE Database" is used with different meanings depending on the context. It may mean the interface software, the logical database or the information in the database. Where there may be doubt, this manual will make clear what is being discussed.

Table of Contents

Introduction
1. Database Objects and Attributes
- **1.1 Object Representation**
- **1.2 Object Types**
- **1.2.1 as-block**
- **1.2.2 as-set**
- **1.2.3 aut-num**
- **1.2.4 domain**
- **1.2.5 filter-set**
- **1.2.6 inet6num**
- **1.2.7 inetnum**
- **1.2.8 inet-rtr**

1.2.9 irt
1.2.10 key-cert
1.2.11 mntner
1.2.12 organisation
1.2.13 peering-set
1.2.14 person
1.2.15 poem
1.2.16 poetic-form
1.2.17 role
1.2.18 route6
1.2.19 route
1.2.20 route-set
1.2.21 rtr-set
2. Updates in the RIPE Database
2.1 Email Update Method
2.2 Synchronous Update Method
2.3 New Starter Update Method
2.4 Format of an Update Message
2.5 Creating, Modifying and Deleting an Object
2.5.1 Object Processing
2.5.2 Creating a New Object
2.5.2.1 Creating person and role Objects
2.5.2.2 Creating organisation Objects
2.5.3 Modifying an Existing Object
2.5.4 Deleting an Object
2.5.5 Garbage Collection
2.5.6 Special Considerations for person and role Objects
2.5.6.1 Re-use of NIC Handles
2.5.6.2 White Pages
2.6 Email Updates
2.6.1 MIME Support
2.6.2 PGP and X.509 Support
2.6.3 Subject Line Processing
2.6.3.1 NEW Keyword
2.6.3.2 HELP Keyword
2.7 Acknowledgements and Notifications
2.7.1 Acknowledgements
2.7.2 Notifications
2.8 Protecting Data
2.8.1 Authorisation Model
2.8.2 Protection of Individual Objects
2.8.3 Protection of person and role Objects
2.8.4 Protection of aut-num Object Space
2.8.5 Protection of Address Space
2.8.6 Protection of Route Object Space
2.8.6.1 Pending Authentication

2.8.7 Protection of Objects with Hierarchical Names
2.8.8 Protection of domain Object Space
2.8.9 Protecting Membership of a Set
2.8.10 Referencing an irt Object
2.8.11 Referencing an organisation Object
2.8.12 Reclaim Functionality
3. Using the RIPE Database Efficiently
3.1 Using the role Object
3.2 Using the organisation Object
3.3 Abuse Handling
Appendices
A1. Object Attributes
A2. Copyright Information
A2.1 RIPE Database Copyright
A2.2 RIPE NCC Copyright
Acknowledgements
References

Introduction

The RIPE Network Management Database (often called the "RIPE Database") contains information about IP address space allocations and assignments, routing policies, reverse delegations and contacts in the RIPE NCC service region [16].

While the information in the RIPE Database is made freely available to the public, it is subject to Terms and Conditions [25].

This document focuses on how to update the database. It does not explain in any detail how the database software works. It does not always make clear why updates work this way. For that you may need to refer to RIPE Policy documents [19] This document is not a statement of any RIPE Policy.

This document is self-contained, but does not provide many examples of usage or illustrations of how the RIPE Database works. The "RIPE Database User Manual: Getting Started" [5] contains some examples. You may also need to read the "RIPE Database Query Reference Manual" [18], which explains how to query the RIPE Database. The query and update manuals form a complimentary set.

1. Database Objects and Attributes

The RIPE Database contains records of:

- Allocations and assignments of IP address space (the IP address registry or INR);
- Reverse domain registrations;
- Routing policy information (the Internet Routing Registry or IRR);

- Contact information (details of people who are registered as the contacts for the Internet resources used in the operation of networks or routers, and their organisations).

The RIPE NCC maintains the Internet resources in the database that are allocated or assigned by the RIPE NCC. These resource objects are tagged as 'RIPE-REGISTRY-RESOURCE'. For other resource objects tagged as 'RIPE-USER-RESOURCE', the registered contacts (Registrants) and the holders of the referenced **mntner** objects maintain this data.

1.1 Object Representation

The records in the RIPE Database are known as **"objects"**. RPSL defines the syntax of database objects. For further information see RFC2622 [1]. An object belongs to one of these object types.

Table 1.1 Object Types Supported by the RIPE Database

Object type (Class name)	Short name	Description
as-block	Ak	Delegation of a range of Autonomous System (AS) Numbers to a given repository.
as-set	As	Set of **aut-num** objects.
aut-num	An	AS Number. It can describe the external routing policy of the AS.
domain	dn	Reverse domain registrations.
filter-set	fs	Set of routes matched by its filter.
inet6num	i6	Allocations and assignments of IPv6 address space.
inetnum	in	Allocations and assignments of IPv4 address space.
inet-rtr	ir	Internet router.
irt	it	Contact and authentication information about a Computer Security Incidence Response Team (CSIRT).
key-cert	kc	Public key certificate that is stored on the server to be used with a **mntner** object for authentication when performing updates.
mntner	mt	Authentication information needed to authorise creation, deletion or modification of the objects

		protected by the **mntner**.
organisation	oa	Organisation that holds the resources.
peering-set	ps	Set of peerings.
person	pn	Technical, administrative and DNS zone contacts.
poem	po	Humorous poem.
poetic-form	pf	Type of humour for a **poem** object.
role	ro	Technical, administrative, DNS zone and abuse contacts - describes a role performed by one or more people.
route	rt	IPv4 route advertised on the Internet.
route6	r6	IPv6 route advertised on the Internet.
route-set	rs	Set of routes.
rtr-set	is	Set of routers.

The RIPE NCC defines a database object as a list of attribute-value pairs in plain text form. The database software uses characters with Latin1 encoding. If you use any other character sets, or non-printable characters, it may cause problems and your updates could fail. There may be a move to UTF-8 soon.

When an object is stored in the database, the attributes and values are unchanged and the order is maintained. The spacing between lines may be adjusted to align the values and make them easier to read.

Each attribute-value pair must start on a separate line. A blank line marks the end of an object (\n\n). This is why you cannot include a completely blank line in the middle of an object.

Attribute names have a precisely defined syntax and only use alpha numeric and the hyphen (-) characters. They are not case sensitive, but most people use lower case. The attribute name must start at column 0 and must immediately be followed by a colon (:). No spaces or tabs are allowed in between the attribute name and the colon. For example:

mntner: TEST-DBM-MNT

The first attribute must have the same name as the object type. This identifies the object type. If the object type is not recognised, that part of the update message will be treated as 'other' text and will be disregarded. Other attributes can appear in any order, but most people stick to the order as shown in the object templates. Each object is uniquely identified by a set of possible attributes. For most object types the primary key is normally the value of the first attribute. In some cases, it is a different

attribute value or a composite of more than one attribute value. The attributes that make up the object are shown in the object templates along with the primary key.

The value part of the attribute-value pair starts after the colon (:). It can contain some pre-defined keywords, references to other objects and free text. You can refer to other objects by using their 'primary key' values. These references and the keywords have a precisely defined syntax. If the objects you refer to do not already exist in the database, you will see error messages and your update will fail. The free text has no syntax, but may only contain Latin1 recognisable characters.

Attribute values may contain spaces and tab characters to help make the information easier to read. Note that spaces work better than tabs, as tabs can display differently on different machines.

You can also split a value over several lines by placing a space, a tab or a plus (+) sign in column 0 of each of the continuation lines. If you wish, you can use more spaces after the continuation character to make it easier to read. The plus (+) sign for line continuation allows attribute values to contain an almost blank line. For example:

address: Street 5

+ City

A blank line marks the end of an object. This is why you cannot include a completely blank line in the middle of an object.

An attribute value may also contain 'end of line' comments. These start with a hash (#) and continue until the end of the physical line. If a value is split over several lines, any of the lines may include an 'end of line' comment. You cannot continue the comment on another line. These comments always stop at the end of the line in which they start. An end of line comment cannot start at column 0. It is possible to add end of line comments on several consecutive lines, which together form a block of text. However, for long comments, it is better to use the "remarks:" attribute.

Any free form value cannot contain a hash (#) as this would be treated this as a comment. Although the software does not process comments, in some situations it does strip off the comments before using the values.

Long end of line comments or long free form attribute values can cause problems. Some mail clients automatically break lines at a certain point. If your mail client does this on an update message then your update will fail. It may not be obvious at first sight that this has happened.

Attributes can be mandatory, optional or generated. The object template shows the type of each attribute per object. They can be listed using the query:

whois –t <object-type>

There is also a more verbose description available by querying:

whois –v <object-type>

You MUST define 'mandatory' attributes in all instances of an object type

You can leave out 'optional' attributes. However, if you choose to include them, then both the attribute and its value must be syntactically correct. Optional attributes are sometimes 'required' by the business rules enforced by the software.

Any 'generated' attributes are created if you leave them out. Where a user-supplied value is not correct, the software will replace it with a new generated value.

When you leave out an optional or generated attribute, remove it completely from the object. You cannot include the attribute name and leave the value blank (unless it is free text).

If the software changes a user-supplied value, it will explain the change in a warning message returned to the user.

Attributes can have single or multiple values.

You can only include a 'single valued' attribute once in an object and it cannot have a list of values.

You can include a 'multiple valued' attribute many times within an object. Some of these attribute instances may also have lists of values separated by commas.

Only attributes with free text values (no keywords or references to other objects) can have a blank value. The software will treat any other attribute, including optional and generated attributes, with a blank value as a syntax error.

All attribute values have a defined syntax. For a detailed description of the attributes supported in the RIPE Database, you should refer to Appendix A1, 'Object Attributes'.

1.2 Object Types

This section describes the object types that the RIPE Database supports. Some description is included about the definition and use of the attributes of each object type.

A set of object templates show which attributes are allowed in each object type. We use the following definitions in the templates:

[mandatory]	You must include at least one instance of this attribute in an object of this type.
[optional]	This attribute is optional in the objects of this type and you can leave it out, unless it is 'required' by the software business rules.

[generated]	These values are generated if you leave them out. If you provide the value it may be replaced.
[single]	Objects must not contain more than one instance of this attribute - value pair.
[multiple]	Objects may contain more than one instance of this attribute. For some attributes, an instance may contain more than one value, separated by commas.
[lookup key]	This attribute is indexed.
[inverse key]	This attribute is in the "reverse" index.
[primary key]	This attribute is (part of) the primary key for this object type.
[primary/lookup key]	This attribute is indexed and is also (part of) the primary key.

In an object template the first column represents an attribute., Tthe second and third columns specify the type of the attribute. and tThe fourth column tells whether the attribute is (part of) a database key for the object.

The "changed:" and "source:" attributes are mandatory in all objects. There must be at least one “changed:” attribute. The information in this attribute may show who created or modified the object and when. It is not reliable as a full audit trail. It is used as a reference for the benefit of the maintainer of the object. It is not intended to give any reliable information to a user who queries for an object.

- The "changed:" attribute must contain an email address and a timestamp. If the timestamp is not included the database software will add the current timestamp at the time of the update.
- The dates can be set to any date after April 1984.
- Any “changed:” attribute can be modified or deleted by the user as long as at least one remains.
- The "source:" attribute specifies the registry where the object is registered. This should be "RIPE" for the RIPE Database. “TEST” is also a valid “source:” attribute, it indicates that the object belongs to the RIPE TEST database.
- The RIPE Database is configured to not allow updates to any other source than RIPE or TEST.

All objects must be maintained. Therefore the “mnt-by:” attribute is mandatory in all objects.

1.2.1 as-block

Only the RIPE Database Administrators can create **as-block** objects.

An **as-block** object delegates a range of AS Numbers to a given RIR [28].

This object sets the authorisation required for the creation of **aut-num** objects within the range specified by the "as-block:" attribute. This is set by the "mnt-lower:" and "mnt-by:" attributes.

1.2.2 as-set

An **as-set** object defines a set of **aut-num** objects.

- "as-set:" - defines the name of the set. It is an RPSL name that starts with "as-". It can be hierarchical. A hierarchical **as-set** name is a sequence of **as-set** names and AS Numbers separated by colons. At least one component must be an actual as-set name (i.e. start with "as-"). All the set name components of a hierarchical as-name have to be as-set names.
- "members:" - lists the members of the set. It can be either a list of AS Numbers, or other as-set names.
- "mbrs-by-ref:" - can be used in all "set" objects; it allows indirect population of a set. If this attribute is used, the set also includes objects of the corresponding type (**aut-num** objects for as-set, for example) that are protected by one of these maintainers and whose "member-of:" attributes refer to the name of the set. If the value of a "mbrs-by-ref:" attribute is ANY, any object of the corresponding type referring to the set is a member of the set. If the "mbrs-by-ref:" attribute is missing, the set is defined explicitly by the "members:" attribute.
- This object sets the authorisation required for the creation of other **as-set** objects one level down in a hierarchy. This is set by the "mnt-lower:" and "mnt-by:" attributes.

1.2.3 aut-num

The **aut-num** object contains registration details of an AS Number and allows routing policies to be published. It refers to a group of IP networks that have a single and clearly- defined external routing policy, operated by one or more network operators – an Autonomous System (AS).

These are 32 bit numbers. Historically, these were 16-bit numbers. But now those are now just a subset of the larger 32-bit set of numbers. The primary key values will be in this format:

ASn where n is a 32-bit number

Leading zeroes (AS0352) are not allowed and will be removed (AS352) by the database software.

- "aut-num:" – the registered AS Number of the Autonomous System that this object describes and starts with AS.
- "as-name:" - a symbolic name of the AS.
- "member-of:" - identifies a set object that this object wants to be a member of. This claim, however, should be acknowledged by a respective "mbrs-by-ref:" attribute in the referenced object.
- "import:", "export:" and "default:" - specify the routing policies of the AS.
- "import-via:" and "export-via:" - specify routing policies regarding non-adjacent networks. These attributes help participants of Multi-Lateral Peering services to inform the intermediate autonomous system what routing policy should be applied towards other participants. Please refer to [33] for more information.
- "mp-import:", "mp-export:" and "mp-default:" - specify IPv4, IPv6 and multicast routing policies.
- "org:" -– single- valued to make sure that only one organisation is responsible for this resource.
- This object sets part of the authorisation required for the creation of **route** and **route6** objects. This is set by the "mnt-routes:", "mnt-lower:" and "mnt-by:" attributes.

1.2.4 domain

The **domain** object represents reverse DNS delegations.

- You should write the domain name in fully qualified format, with or without a trailing dot. If a trailing dot is included, it will be removed by the software and a warning message returned to the user.
- If the nameserver name in the "nserver:" attribute is inside the domain being delegated it may be optionally followed by an IP address (IPv4 or IPv6).
- "ds-rdata:" - holds information about a signed delegation record for DNSSEC (short for DNS Security Extensions)

1.2.5 filter-set

A **filter-set** object defines a set of routes that match the criteria that you specify in your 'filter' – in other words, it filters out routes that you do not want to see.

- "filter-set:" - defines the name of your filter. It is an RPSL name that starts with "fltr-". It can be hierarchical. A hierarchical filter-set name is a sequence of filter-set names and AS Numbers separated by colons. At least one component of the

name must be an actual filter-set name (i.e. start with "fltr-"). All the set name components of a hierarchical filter-name have to be filter-set names.

- "filter:" - (optional) defines the policy filter of the set. This is a logical expression that, when applied to a set of routes, returns a subset of these routes. These are the ones that you want to filter in or out.
- "mp-filter:" - (optional) extends the "filter:" attribute to allow you to specify IPv4 or IPv6 prefixes and prefix ranges.
- "filter:" and "mp-filter:" - at least one of these two attributes is required but not both within the same object.
- This object sets the authorisation required for the creation of other **filter-set** objects one level down in a hierarchy. This is set by the "mnt-lower:" and "mnt-by:" attributes.
- Unlike other set objects this one does not include the "mbrs-by-ref:" attribute.

1.2.6 inet6num

An **inet6num** object contains information on allocations and assignments of IPv6 address space.

- "inet6num:" - specifies a range of IPv6 addresses that the **inet6num** object presents. The range may be a single address. Addresses can only be expressed in prefix notation
- The prefix notation expresses addresses in hexadecimal groups of two bytes separated by colons and with the possible use of shorthand notation for strings of consecutive zeros. Leading zeros from any two-byte group will be removed by the software. In this case a warning message will be returned to the user.
- "netname:" - is the name of a range of IP address space. It is recommended that the same netname is used for any set of assignment ranges used for a common purpose.
- "country:" - identifies a country. It has never been specified what this country means. It cannot therefore be used in any reliable way to map IP addresses to countries.
- "org:" - single value to ensure that only one organisation is responsible for this resource.
- "status:" - an administrative tag to register the type of address space.
- Status can have one of these values:
    - ALLOCATED-BY-RIR
    - ALLOCATED-BY-LIR
    - AGGREGATED-BY-LIR

    - ASSIGNED
    - ASSIGNED ANYCAST
    - ASSIGNED PI
- The **inet6num** object sets the authorisation required for the creation of more specific **inet6num** objects within the range specified by this **inet6num**. This is set by the "mnt-lower:" and "mnt-by:" attributes.
- This object sets the authorisation required for the creation of a **domain** object for reverse delegation. This is set by the "mnt-domains:", "mnt-lower:" and "mnt-by:" attributes.
- This object sets part of the authorisation required for the creation of a **route6** object. This is set by the "mnt-routes:", "mnt-lower:" and "mnt-by:" attributes.
- "mnt-irt:" - references an **irt** object. Authorisation is required from the **irt** object to be able to add this reference.

1.2.7 inetnum

An **inetnum** object contains information on allocations and assignments of IPv4 address space.

- "inetnum:" - specifies a range of IPv4 addresses that the **inetnum** object presents. The range may be a single address. If the range represents multiple addresses, the ending address should be greater than the starting one.
- Addresses can be expressed in either range or prefix notation. If prefix notation is used, the software will convert this to range notation and a warning message will be returned to the user.
- The range notation expresses addresses as 32-bit whole numbers in dotted quad notation. Leading zeros from any quad will be removed by the software and a warning message will be returned to the user.
- "netname:" - is the name of a range of IP address space. It is recommended that the same netname be used for any set of assignment ranges used for a common purpose, such as a customer or service.
- "country:" - identifies the country. It has not been specified what this country means. It cannot therefore be used in any reliable way to map IP addresses to countries.
- "org:" - single value to ensure only one organisation is responsible for this resource.
- "status:" - an administrative tag to register the type of address space.
- Status can have one of these values:
    - ALLOCATED UNSPECIFIED

    - ALLOCATED PA
    - ALLOCATED PI
    - LIR-PARTITIONED PA
    - LIR-PARTITIONED PI
    - SUB-ALLOCATED PA
    - ASSIGNED PA
    - ASSIGNED PI
    - ASSIGNED ANYCAST
    - EARLY-REGISTRATION
    - NOT-SET
- The **inetnum** object sets the authorisation required for the creation of more specific **inetnum** objects within the range specified by this **inetnum**. This is set by the "mnt-lower:" and "mnt-by:" attributes.
- This object sets the authorisation required for the creation of a **domain** object for reverse delegation. This is set by the "mnt-domains:", "mnt-lower:" and "mnt-by:" attributes.
- This object sets part of the authorisation required for the creation of a **route** object. This is set by the "mnt-routes:", "mnt-lower:" and "mnt-by:" attributes.
- The "mnt-irt:" attribute references an **irt** object. Authorisation is required from the **irt** object to be able to add this reference.

1.2.8 inet-rtr

The **inet-rtr** object specifies routers.

- "inet-rtr:" - a valid DNS name for a router without a trailing dot.
- "alias:" - (optional) standard DNS name for the specified router.
- "local-as:" - specifies the AS Number of the AS that owns or operates this router.
- "ifaddr:" - specifies the interface address within an Internet router, as well as an optional action to set other parameters on this interface.
- "interface:" - specifies a multi-protocol interface address within an Internet router, optional action and tunnel definition.
- "peer:" - specifies the details of any interior or exterior router peering.
- "mp-peer:" - extends the "peer:" attribute to support both IPv4 and IPv6 addresses.
- "member-of:" - identifies a set object that this object wants to be a member of. This claim, however, should be acknowledged by a respective "mbrs-by-ref:" attribute in the referenced object.

1.2.9 irt

An **irt** object represents a Computer Security Incident Response Team (CSIRT). It includes and security information. It may be referenced from **inetnum** or **inet6num** objects to show which CSIRT is responsible for handling computer and network incidents for that address range.

- **irt** - name starts with "IRT-".
- "signature:" - references a key-cert object representing a CSIRT public key used by the team to sign their correspondence.
- "encryption:" - references a **key-cert** object representing a CSIRT public key used to encrypt correspondence sent to the CSIRT.
- "auth:" - defines an authentication scheme to be used. Any of the current authentication schemes used by the RIPE Database are allowed.
- To add a reference to an **irt** in an **inetnum** or **inet6num** object the authorisation must be passed from one of the "auth:" values in the referenced **irt** object.
- "irt-nfy:" - specifies the email address to be notified when a reference to the **irt** object is added or removed.

1.2.10 key-cert

A **key-cert** object is a database public key certificate that is stored in the RIPE Database. It is used with a **mntner** object for authentication when performing updates. Currently the RIPE Database supports two types of keys.

- For PGP **key-cert** objects, the value of the "key-cert:" attribute must be PGP-"key-id". These keys are compliant with the Open PGP Message Format [23].
- For X.509 **key-cert** objects, the database software assigns this value as X.509-n. Here, 'n' is the next available number assigned by the software. If you want to create an X.509 **key-cert** object, you should specify the value as AUTO-xx. If you delete an X.509 **key-cert** object, it is not possible to recreate it with the same name.
- "method:", "owner:" and "fingerpr:" - all generated by the software. It is not necessary to include these attributes when you create or modify this object. If they are supplied, the software will check the values. If necessary the software will replace the supplied values with generated values. In this case a warning is returned to the user.
- "certif:" - contains the public key. All the lines of an exported key in ASCII armoured format must be included, as well as the start/end markers and the empty line which separates the header from the key body.

1.2.11 mntner

Objects in the RIPE Database are protected by using **mntner** objects. A **mntner** object is an anonymous box containing the information needed to authorise creation, deletion or modification of any objects that it protects by whoever maintains this data.

- Objects are protected by a **mntner**, if they contain a reference to the **mntner** in the object. This is done by including a "mnt-by:" attribute. Other attributes offer hierarchical protection. The "mnt-by:" attribute is mandatory in all object types. Most users set the "mnt-by:" value in a **mntner** to reference itself.
- "referral-by:" - can refer to the **mntner** object itself. The database software does not use this attribute, even though it is mandatory to include it.
- "upd-to:" - specifies the email address to be notified when an attempt to update an object protected by this mntner is unsuccessful due to authorisation failure.
- "mnt-nfy:" - specifies the email address to be notified when an object protected by this mntner is successfully updated.
- "auth:" - defines an authentication scheme to be used. Any of the current authentication schemes used by the RIPE Database are allowed.
- To update an object protected by a **mntner** the authorisation must be passed from one of the "auth:" values in the **mntner** object referenced in one of the "mnt-by:" attributes of the updated object. This means the correct credential for one of the "auth:" values must be supplied as part of the update.
- If an object references more than one **mntner** in the "mnt-by:" attributes they act as a logical 'OR'. If the authorisation is passed by any "auth:" value from any of the referenced **mntner** objects then the update will be authorised.
- The "mnt-lower:", "mnt-routes:" and "mnt-domains:" attributes all provide for hierarchical authorisation. These also work in a logical 'OR' when multiple values are included in an object. How they are used is described in the object descriptions where these attributes are valid.

1.2.12 organisation

The **organisation** object provides information about an organisation that has registered a network resource in the RIPE Database. This may be a company, non-profit group or individual. It was introduced as a means to link together all the human and Internet resources related to one organisation.

- "organisation:" - specifies the ID of the **organisation** object. This is set by the database software. It is used as a reference label. When creating an organisation

this should be set to "AUTO-n-<optional letter Combination>", where n is a number greater than 0.

- "org-name:" - defines the name of the organisation.
- "org-type:" - specifies the type of an organisation and can be one of these:
  - IANA
  - RIR
  - NIR (Note - there are no NIRs in the RIPE NCC service region.)
  - LIR
  - WHITEPAGES
  - DIRECT_ASSIGNMENT
  - OTHER
- Users can only set the "org-type:" to OTHER. All remaining types are only for use by the RIPE Database aAdministrators.
- "org:" - used to reference an **organisation** object for an organisation that has registered the resource or other data in which this reference is made.
- "ref-nfy:" - specifies the email address to be notified when a reference to the **organisation** object is added or removed.
- "mnt-ref:" - specifies the maintainer objects that are allowed to authorise the addition of references to the **organisation** object in other objects.
- If the **organisation** object includes more than one "mnt-ref:" attribute they act as a logical 'OR'. If the authorisation is passed by any **mntner** object referenced in any "mnt-ref:" attribute then the update will be authorised.
- The "abuse-c:" attribute specifies the id (nic-hdl) of a **role** object holding contact information for abuse complaints. When this attribute is specified no other email address should be used for abuse complaints.

1.2.13 peering-set

A **peering-set** object defines the set of peerings that appear in the "peering:" or "mp-peering:" attribute.

- "peering-set:" - defines the name of the set. It is an RPSL name that starts with 'prng-'. It can be hierarchical. A hierarchical peering-set name is a sequence of peering-set names and AS Numbers separated by colons. At least one part of the name must be an actual **peering-set** name (i.e. start with "prng-"). All the set name components of a hierarchical **peering-set** have to be **peering-set** names.
- "peering:" - (optional) defines a peering that you can use to import or export routes.

- "mp-peering:" - (optional) extends the "peering:" attribute and defines a multiprotocol peering that can be used to import or export both IPv4 and IPv6 routes.
- "peering:" and "mp-peering:" - at least one of these two attributes is required but not both in the same object.
- This object sets the authorisation required for the creation of other **peering-set** objects one level down in a hierarchy. This is set by the "mnt-lower:" and "mnt-by:" attributes.

1.2.14 person

A **person** object contains information about contact(s) responsible for an object. After it has been created, the "person:" attribute cannot be changed.

The person object is identified by the "nic-hdl:" attribute. This is a label, usually made up from the initials of the person's name and the database "source:" (for example, DW-RIPE).

- "nic-hdl:" - usually made up from the initials of the person's name and the database "source:" (for example, DW-RIPE). It can use an international country code instead of the database source as the suffix or omit the suffix (for example DW-NL or just DW). It is used by other objects to reference the person.
- The user can specify the "nic-hdl:" when the object is created or can use the "AUTO-n <optional letter combination>" construction, where n is any number greater than 0.Table 1.1, 'Object Types Supported by the RIPE Database'Table 1.1, 'Object Types Supported by the RIPE Database'
- The "nic-hdl:" attributes of the **person** and **role** objects share the same name space in the database. So you cannot create a **person** and **role** using the same "nic-hdl:". The "nic-hdl:" must be unique across both object types.
- "address:" - is the full postal address of the contact in free format text.
- "phone:" - specifies a telephone number of the contact in international shorthand. It must start with a '+' followed by the international country code, area code and number.
- "fax-no:" - has the same format as the "phone:" attribute.
- "e-mail:" - represents the contact's email address as defined by RFC 2822 [8].
- "remarks:" - can be any free format text.
- "notify:" - specifies the email address to which notifications of changes to an object should be sent.
- "mnt-by:" - specifies the identifier of a registered **mntner** object used for authorisation of operations performed on the object that contains this attribute. The **mntner** object must exist in the database before it can be referenced in other objects.

1.2.15 poem

A **poem** object contains a poem that is submitted by a user. It has no operational use and reflects the humorous side of the industry representatives.

- "poem:" - specifies the title of the poem.
- "form:" - specifies the identifier of a registered poem type.
- "text:" - specifies the body of the poem. It must be humorous, but not malicious or insulting. It should be written in the style of the "form:".
- "author:" - is the "nic-hdl:" of the person who entered the poem.

1.2.16 poetic-form

A **poetic-form** object defines the supported poem types.

- "poetic-form:" - starts with "FORM-". It is followed by the name of an internationally recognised poetic format of humorous writing. For example, limerick or haiku.
- "descr:" - describes the style of the poetic form, written in the form style. For example, if it is a FORM-LIMERICK, the description will be written as a limerick.

1.2.17 role

A **role** object is similar to a **person** object. However, instead of describing a single person, it describes a role performed by one or more people. This might be a help desk, network monitoring centre, system administrator etc. A **role** object is useful since often a person performing a specific function may change while the role itself remains.

- If one person needs to be referenced in a large number of objects, it is better to use a **role** object and then place that person in the **role** object. If that person leaves your company, it is simple to modify the **role** object.
- The **role** object is identified by the "nic-hdl:" attribute. This is a label, usually made up from the initials of the function and the database "source:" (for example, CREW-RIPE).
- The "nic-hdl:" can use an international country code instead of the database source as the suffix or omit the suffix (for example CREW-NL or just CREW).
- The user can specify the "nic-hdl:" when the object is created or can use the "AUTO-n <optional letter combination>" construction, where n is any number greater than 0.
- "nic-hdl:" attributes of the **person** and **role** objects share the same name space in the database. You cannot create a **person** and **role** using the same "nic-hdl:". The "nic-hdl:" must be unique across both object types.

- The "abuse-mailbox:" attribute specifies the email address to which abuse complaints should be sent. When this attribute is specified no other email address should be used for abuse complaints.

1.2.18 route6

Each interAS route (also known as an interdomain route) originated by an Autonomous System can be specified by using a **route6** object for IPv6 addresses

It is possible to create **route6** objects in the RIPE Database for address space or AS Numbers registered in other RIR regions.

- "route6:" - IPv6 address prefix of the route.
- "origin:" - AS Number of the Autonomous System that originates the route into the interAS routing system. The corresponding **aut-num** object for this Autonomous System must already exist in the RIPE Database.
- "route6:" and "origin:" - pair of attributes that make up the primary key.

1.2.19 route

Each interAS route (also known as an interdomain route) originated by an Autonomous System can be specified by using a **route** object for IPv4 addresses.

It is possible to create **route** objects in the RIPE Database for address space or AS Numbers registered in other RIR regions.

- "route:" - address prefix of the route.
- "origin:" - AS Number of the Autonomous System that originates the route into the interAS routing system. The corresponding **aut-num** object for this Autonomous System must already exist in the RIPE Database.
- "route:" and "origin:" - pair of attributes that make up the primary key.
- "holes:" - list the component address prefixes that are not reachable through the aggregate route (perhaps that part of the address space is unallocated).
- "member-of:" - can be used in the **route**, **route6**, **aut-num** and **inet-rtr** classes. The value of the "member-of:" attribute identifies a set object that this object wants to be a member of. This claim, however, should be acknowledged by a respective "mbrs-by-ref:" attribute in the referenced object.
- "inject:" - specifies which routers perform the aggregation and when they perform it.
- "aggr-mtd:" - specifies how the aggregate is generated.
- "aggr-bndry:" - defines a set of Autonomous Systems, which form the aggregation boundary.

- "export-comps:" - defines the set's policy filter, a logical expression which when applied to a set of routes returns a subset of these routes.
- "components:" - defines what component routes are used to form the aggregate.
- This object sets the authorisation required for the creation of more specific **route** objects within the range specified by this **route**. This is set by the "mnt-lower:" and "mnt-by:" attributes.
- This object sets part of the authorisation required for the creation of a more specific **route** object. This is set by the "mnt-routes:", "mnt-lower:" and "mnt-by:" attributes.
- The "mnt-routes:" attribute can include an optional list of prefix ranges inside of curly braces ("{}") or the keyword "ANY". This should follow after the reference to the maintainer. The default, when no additional set items are specified, is "ANY" or all more specifics. Please refer to RFC-2622 [1] for more information.

1.2.20 route-set

A **route-set** object is a set of route prefixes and not a set of database **route** objects.

- "route-set:" - defines the name of the set. It is an RPSL name that starts with "rs-". It defines a set of routes that can be represented by **route** objects or by address prefixes. It can be hierarchical. A hierarchical route-set name is a sequence of route-set names and AS numbers separated by colons. At least one component of such a name must be an actual route-set name (i.e. start with "rs-").
- In the case of **route** objects, the set is populated by means of the "mbrs-by-ref:" attribute. In the case of address prefixes, the members of the set are explicitly listed in the "members:" attribute.
- "members:" - list of address prefixes or other route-set names.
- "mp-members:" - list of IPv4 or IPv6 address prefixes or other route-set names.
- "mbrs-by-ref:" - can be used in all "set" objects. It allows indirect population of a set. If this attribute is used, the set also includes objects of the corresponding type (**aut-num** objects for as-set, for example) that are protected by one of these maintainers and whose "member-of:" attributes refer to the name of the set. If the value of a "mbrs-by-ref:" attribute is ANY, any object of the corresponding type referring to the set is a member of the set. If the "mbrs-by-ref:" attribute is missing, the set is defined explicitly by the "members:" attribute.
- This object sets the authorisation required for the creation of other **route-set** objects one level down in a hierarchy. This is set by the "mnt-lower:" and "mnt-by:" attributes.

1.2.21 rtr-set

A **rtr-set** object defines a set of routers.

- A set may be described by the "members:" attribute that is a list of inet-rtr names, IPv4 addresses or other rtr-set names. The "mp-members:" attribute extends the "members:" attribute to use IPv6 addresses.
- A set may also be populated by means of the "mbrs-by-ref:" attribute, in which case it is represented by inet-rtr objects.
- "rtr-set:" - defines the name of the set. It is an RPSL name that starts with "rtrs-". It can be hierarchical. A hierarchical rtr-set name is a sequence of rtr-set names and AS Numbers separated by colons. At least one component of such a name must be an actual rtr-set name (i.e. start with "rtrs-").
- This object sets the authorisation required for the creation of other rtr-set objects one level down in a hierarchy. This is set by the "mnt-lower:" and "mnt-by:" attributes.

2. Updates in the RIPE Database

To create, modify or delete an object you need to send an update message to the database.

There are two databases that you can update. The RIPE Database is the authoritative database and is sometimes referred to as the live or production database. There is also a RIPE TEST Database. This operates in the same way but contains only test data. The test data is cleaned out every night and a pre-determined, basic set of objects is re-inserted. The TEST Database can be used to learn how to update the database and try out 'what if' scenarios. There are fewer restrictions allowing you to create encompassing or parent objects that you may need for your tests.

You can submit most updates to either database using three access methods

- By sending an email.
- By using the REST APi.
- By using the synchronous update service (often referred to as syncupdates). There are two ways to do this:
  - By using a client to access the "syncupdates" service.
  - By using the "webupdates" form from the www.ripe.net website [22]. This is an interface to the syncupdates service.

2.1 Email Uupdate Mmethod

Email update messages must be in plain text and can be MIME encoded. If encoded, the database will treat each valid MIME part as a separate message. But, where messages are multipart/signed, the signature part will be associated with the corresponding MIME text part. Please see Section 2.6.1, 'MIME Support' for more information. An update message, or MIME part, may contain more than one object.

To submit an email update to the RIPE Database, send an email to auto-dbm@ripe.net. To submit an email update to the TEST Database, send an email to test-dbm@ripe.net.

Acknowledgements from the database are returned to the sender of the email based on the "Reply-to:" and "From:" fields in the email header. Notifications may also be sent based on email addresses found in the attributes within each object in the update message and the related **mntner** objects.

There are no restrictions on the number of objects in any one email message. But there is a limit on the total size of an update message. It is up to the user to determine how many objects to include in an update message. There are some points to consider.

- Every email update message will result in one acknowledgement email message returned.
- Each object in an update message may generate several notification email messages.
- If you update 1,000 objects by sending 1,000 emails, each containing one object, this will cause a large number of emails to be returned. Your mail system needs to be able to handle the volume of emails generated in this way. Because of the volume of emails from one source, it may also be seen as spam by some mail agents.
- The acknowledgement and notifications are not sent until all the objects in an update message have been processed.
- If you submit one update message containing 1,000 objects, no response will be received until all the objects have been processed.
- You need to optimise the number of update messages and number of objects per update to suit your working practises.
- Occasionally an update message causes system problems. The Database Administrator will monitor the progress of such an update.

2.2 Synchronous Uupdate Mmethod

The synchronous update method allows near instant updating of the RIPE Database. It is designed for applications that need to update the RIPE Database and see an immediate change. It returns the response header and an acknowledgement (if any)

in text/plain format. See the separate document for details of the protocol for communicating with syncupdates [20].

There is no maximum amount of time for an update operation to be completed. You should set your timeout to the highest possible value. Problems sometimes occur with reverse **domain** objects. Because of all the DNS checks made, reverse **domain** objects sometimes take longer to process, and your connection may occasionally timeout. The update will still be completed by the database server, but no acknowledgement will be returned if the connection has timed out. Notification messages are always sent by email, and will be sent regardless of your connection status.

There are sample clients to use with syncupdates which that access either the RIPE Database or the TEST Database. Although they are sample clients they can be used 'as is' from our web site [21]. There is also a short perl script which you can use as a client to access syncupdates [21].

You can enter the objects directly into the text area of this web form. There is no limit to the number of objects you can enter in one submission. But the same principles apply as described in Section 2.0.1, 'E-Mail Update Method'. The main processing of an update message is the same for any method of submission. So the rules about responses and notification emails are the same. With syncupdates, if you enter many objects your client connection may time out waiting for the final response after processing all the objects.

Webupdates can be accessed from the links at http://apps.db.ripe.net/[22]. You can select either the RIPE Database or TEST Database by setting the source. Details of how to use webupdates can be found in the "RIPE Database User Manual-Getting Started" [5].

Webupdates is normally used for individual object updates. Multiple objects can be submitted by switching the view to use the text input area similar to syncupdates. If more than one object is entered into the text area it is then not possible to switch view back to the attribute working mode.

2.3 New Sstarter Uupdate Mmethod

This update method allows a new starter, who has no objects in the RIPE Database, to get started. The first step is to create a **person** and **mntner** objects. But the **person** object must be maintained and the **mntner** object needs to reference personal contacts. So these first two objects reference each other.

This situation cannot be processed directly by the other two standard update methods described above. There is a web form to create these first two objects for a new user [26]

If errors occur during the creation process, these are reported back to the user. If successful, the object identifiers are returned to the user. There is no partial success allowed. If one object can be successfully created during the intermediate stage, but the second object has errors, none will be created.

If you have nothing in the RIPE Database, this startup script is the only way to get started. Once you have the first pair of objects you can use the normal update methods for all other objects.

2.4 Format of an Update Message

The body of an update message is the same regardless of the access method. It contains one or more objects. The structure of the objects is described in Section 1.1, 'Object Representation'.

Object definition starts with the type attribute and ends with the first blank line ("\n\n"). So you cannot include a blank line within an object.

We apply a heuristic method to each paragraph of text in the input to determine if it is an object. Any part of the message that is not recognised as a database object is ignored. These parts are grouped together at the end of the acknowledgement message.

2.5 Creating, Modifying and Deleting an Object

To create, modify or delete objects, you need to send a message to the database by one of the access methods. This message must contain one or more database objects. Table 1.1, 'Object Types Supported by the RIPE Database'lists all the object types that are recognised by the database. No other object types can be created. Each object type is described by templates. They can be listed using the query:

whois –t <object-type>

There is also a detailed description available by querying:

whois –v <object-type>

Each instance of an object must contain at least one of each of the mandatory attributes for that object type. The optional attributes can be left out unless the software business rules require them. One message may contain several objects, with a mixture of creation, modification or deletion operations.

2.5.1 Object Processing

As a rule, the order of objects in the message is not changed. The database software processes objects one by one, starting with the first recognised object in the message. Since the order can matter, the database software tries to figure out the best order of execution. “AUTO-n” attributes to be referenced by other objects in the same update will be resolved, as long as the “n” number is consistent.

You can avoid complex arrangements of auto generated values in a single update message by sending consecutive updates instead. An example of a complex arrangement might be an update containing a **person** object with "nic-hdl: auto-1”,

a **role** object with "nic-hdl: auto-2" and referencing the **person** auto-1 and a **mntner** referencing the **role** object auto-2.

When processing each individual object, the software makes many checks including that:

- The syntax of the object is correct.
- The object passes all required authorisation checks.
- All references to other objects can be resolved without conflicts.
- The operation does not compromise referential integrity. For example, when an object is to be deleted, the server checks that it is not referenced from any other object in the RIPE Database.
- The object complies with all software business rules, for example not creating a **person** object with a nic-hdl already in use by a **role** object.
- The object complies with relevant policies, such as having the correct "status:" value in **inetnum** objects.

If all checks are successful, the update will proceed for this object. If one of these steps fails, the operation fails for this object. This is shown in the acknowledgement message and sometimes in notification messages.

Each object in the update message is processed independently of others, so even if one operation fails, the following objects will still be processed. There may, however, be consequences of a previous failed operation. For example if a **person** object creation fails, a later object creation which that references this **person** object will fail the referential integrity checks because the **person** object does not exist in the database.

After the software finishes processing all the objects in the update message, an acknowledgement message is returned to the sender of the original update. For email update messages, this will be as specified in the "Reply-to:" field or "From:" field. For synchronous update messages it will be returned via the open connection. If the connection has timed out or been closed, no acknowledgement is sent.

Notification messages may also be sent. See Section 2.7.2, 'Notifications' for more information about this.

2.5.2 Creating a NNew Object

If the database does not contain an object of the same type and with the same primary key as the object in the update message, the object will be created (remembering that a **person** and **role** object cannot use the same NIC handle).

2.5.2.1 Creating person and rrole Oobjects

To create **person** and **role** objects, you can use "AUTO NIC handles". If you do this, then the server will automatically assign a NIC handle.

To do this, the value of the "nic-hdl:" attribute should be:

nic-hdl: AUTO-<digit>[<initials>]

If you choose to specify the <initials> (between two and four characters), they will be used to try to make the NIC handle. Otherwise the initials will be taken from the name in the "person:" or "role:" attribute.

The default suffix is "-RIPE" when an "AUTO NIC handle" is used. If you want to use another suffix or have no suffix, or you want to also select the number part as well, then you must specify the full nic-hdl you want. Note that if you specify a nic-hdl that is currently in use it will be considered as a modification operation. This will usually result in an authentication error. If you select a nic-hdl that has been used and deleted an error will be returned.

If you are creating your first **person** object you must follow the procedure described in 2.3, ''New starter update method''.

2.5.2.2 Creating organisation Oobjects

If you want to create an **organisation** object, you must specify the ID of the object as AUTO-<digit>[<initials>]. The database software will then assign an appropriate ID by using the initials of the "org-name:" attribute of the object. If you prefer, you can specify the letter combination you would like to use.

For example, if you want TTR as the letter combination in the organisation ID, you should put “AUTO-1TTR” into the "organisation:" attribute, when you create the object. If you delete an **organisation** object you cannot re-create one with the same ID as the one you deleted.

2.5.3 Modifying an Existing Object

If an object type with the same primary keys as the object in the update message already exists in the database, the assumed operation is object modification. The old and new versions of the object are compared and a no-operation error is reported if they are identical. When comparing the two versions, white space characters are not considered.

2.5.4 Deleting an Object

You can delete an object by adding a "delete:" pseudo attribute to the object.

delete: <comment>

The software will only accept this request if the object in the message is exactly the same as the one in the database which is to be deleted. When comparing the versions, white space characters are not considered. If you query the database for an object to delete so that you get the exact copy of the object, you should use the "-B" query flag. Otherwise you will get a filtered object that will not pass the identical check. The delete operation will fail if the object to be deleted is referenced from any other object in the database.

This pseudo attribute applies to one object only in an update message. It must be a part of the object in the update message that is to be deleted. It can be added at any point within the object or immediately before or after the object..

Objects can still be deleted from the database even if they are not syntactically correct. This allows for old objects to be deleted long after the syntax has been changed.

2.5.5 Garbage Ccollection

When certain object types have existed in the database unreferenced for a continuous period of time it becomes eligible for automatic deletion. When querying for an object it is possible to see if it's eligible for deletion and if so, how long it will remain in the database before deletion. The message "Unreferenced # <object key>' will be deleted in X days" will be returned together with the object if the query-flag --SHOW-TAG-INFO is given.

The object types include **person**, **role**, **mntner**, **organisation** (not of type LIR) and **key-cert**. Clusters of these objects that form a self referencing group, without any reference from resource data, will also be deleted.

2.5.6 Special Cconsiderations for person and role Oobjects

There are a few additional issues that relate specifically to these objects.

2.5.6.1 Re-use of NIC Hhandles

A **person** and **role** object is identified by a NIC handle. Historically, these were available for re-use as soon as the object was deleted. Many NIC handles have been (re-)used by several people over the course of time. Since that can lead to confusion, the NIC handles are no longer re-usable. When a **person** or **role** object is deleted, the NIC handle cannot be re-used. Note that these objects cannot be deleted if they are referenced by any other object. This avoids accidental deletion of an object that is still in use. While some people try to create NIC handles with a '"meaningful'" name, it should be remembered that they are only meant as a database index. If you delete one that is unreferenced, then realise you still need it, you will have to create a new one.

2.5.6.2 White Pages

The RIPE Community recognises that there are some people with a high profile within the Community but who do not manage Internet resources. Some of these people may have a **person** object in the RIPE Database and may use the NIC handle as a signature and for contact purposes. The White Pages is a facility for these people to 'opt in' to having their personal data in the Database. Strict conditions

apply to limit the number of people using this facility [27]. Anyone listed in the White Pages will be exempt from the garbage collection.

2.5.7 Dry-run

The TEST Database has always been promoted as the place to "try" an update to see what happens. But the TEST Database is a sterile land with hardly any data in it. It is also reset every day. So if you want to test anything that has dependencies on other data you have to reproduce it all in the TEST Database to do the test. That in itself can be a major effort. And tomorrow it is all gone.

To address this, there is a "dry-run" feature that lets you test updates on the production RIPE Database. All your data is there. All dependencies are taken care of. Submit your update and see what the result will be. But nothing actually changes.

The details are explained in a RIPE Labs article [32].

2.6 Email Updates

This section describes the way that email messages are processed and the features available with email updates.

2.6.1 MIME Support

The database software supports MIME. This means that you can cryptographically sign an update message using email agents that attach the signature in a separate MIME part, not in the body of the message. However, encryption of the text is not allowed. All update messages must be sent in plain text.

It also allows the definition of scopes of authorisation within the message (for example, parts where different passwords apply) and nested signing of messages. This may be necessary under some conditions when updating objects whose authorisation must be derived from more than one party.

It is **strongly** recommended to keep mime encapsulation simple. Complex mime structures are more likely to generate errors.

The following rules apply when submitting updates using MIME encapsulation:

The software will recognise the following headers and take the appropriate actions:

- multipart/signed
- multipart/alternative
- multipart/mixed
- multipart/unknown
- application/pgp-signature
- application/x-pkcs7-signature

- application/pkcs7-signature
- text/plain

All other content-types are treated as text/plain.

Each MIME part is treated as a separate message with the following implications (except where a signature part is closely coupled with a text part in which case the two parts are treated together as a message):

- Authorisation information is valid only within a single message part
- AUTO NIC handle assignment is made only within a single message part (see Section 2.6.2, 'PGP and X.509 support')

2.6.2 PGP and X.509 Support

The database supports PGP and X.509 signed messages. The following rules apply when submitting updates using these authorisation schemes.

- When using MIME encapsulation a signed portion of an update message should be submitted using multipart/signed composite type. In this case, the first body part contains the update message (which may also be a MIME encapsulated message), and the second body contains a signature. For a PGP signature, it is encapsulated with application/pgp-signature MIME discrete type. For an X.509-signature it is encapsulated with either application/x-pkcs7-signature or application/pkcs7-signature MIME discrete types.

Regarding AUTO NIC handle assignment, the signed portion is treated as a separate message.

If one of the signatures fails in a nested signed portion, the whole portion is rejected.

2.6.3 Subject Line Processing

The subject line can have a special meaning in email update messages by using keywords. The available keywords, which are case insensitive, are:

- NEW
- HELP

One way to use a keyword is to put it in the subject line of the email message, with NO other words present. If any other word is found that is not one of the available keywords (for example, Subject: NEW objects) then none of the words will be treated as keywords. All words in the subject line will be reported in the acknowledgement reply as invalid keywords, along with a WARNING message. In this context it is impossible to know if a word is meant as a keyword or just part of a comment.

Many users often include their own references in the subject line. Using this method it is not possible to also use a keyword. The user's references are reported in the WARNING message as invalid keywords.

Some examples:

Subject: new

This is an accepted keyword.

Subject: sending my new objects

None of these words are accepted as a keyword and all reported in the Warning message.

2.6.3.1 NEW Keyword

Use NEW keyword if you want the database to only accept new objects. In this case, all objects found in the update are assumed to be creation operations. If an object already exists in the database, that object will result in an error message in the acknowledgement.

2.6.3.2 HELP Keyword

The HELP keyword causes a help text to be returned in the acknowledgement that contains information about how to query and update the database. When this keyword is used the body of the update message is ignored.

2.7 Acknowledgements and Notifications

2.7.1 Acknowledgements

One acknowledgement message (ACK) is returned to the user for each update received.

If the update was sent by email, the subject line of the ACK message states "SUCCESS" or "FAILED". If the update message contains no objects or any one of the operations fails, the ACK reports "FAILED". Otherwise it reports "SUCCESS". Following this status is the original subject line. For example:

From: RIPE Database Administration <ripe-dbm@ripe.net>
To: admin@here.com
Subject: SUCCESS: UPDATE person
Reply-To: ripe-dbm@ripe.net

The acknowledgement message content is split into sections.

The first section shows from where the update was received. This may be a copy of the update email header or the IP address for a synchronous connection. This section also includes a summary of the update results, explaining how many objects were recognised by the database software, how many operations were successful and how many failed.

An example first section would look like this:

> From: admin@here.com
> Subject: UPDATE person
> Date: Wed, 28 Mar 2007 01:00:06 +0300 (EEST)
> Reply-To: admin@here.com
> Message-ID: <20070327170006.983D6124225@here.com >

SUMMARY OF UPDATE:

Number of objects found: 2
Number of objects processed successfully: 1
Create: 1
Modify: 1
Delete: 0
No Operation: 0
Number of objects processed with errors: 0
Create: 0
Modify: 0
Delete: 0
Syntax Errors: 0

For a synchronous update the details of where the update was received from would look like this instead of the email headers:

- From-Host: 193.0.0.1
- Date/Time: Wed Mar 28 00:17:29 2007

The next section is the "DETAILED EXPLANATION". This is split into three parts. The first part lists all the objects where the operation failed. This will include "***Error:" messages as well as possible "***Info:" and "***Warning:" messages. The next part shows all the operations that were successful. This may include additional "***Info:" and "***Warning:" messages with each operation listed. The third part lists all the paragraphs from the update message that were not recognised as objects. Each part is separated in the ACK by a line containing several '~' characters. Within each part the objects and paragraphs are listed in the order they were processed. This is generally the order they appear in the update message unless AUTO-n nic-hdls are referenced. The line before each object contains three '-' characters. This allows for easy parsing by script.

An example of this section would look like this:

DETAILED EXPLANATION:

~~~~~~
The following object(s) were found to have ERRORS:

Update FAILED: [person] dn1172-ripe   de nis
****Error: Syntax error in object*

person: de nis
address: here
phone: 1
****Error: Syntax error in "1"*
nic-hdl: dn1172-ripe
mnt-by: TEST-MNT
changed: admin@here.com 20070327
source: ripe

~~~~~~
The following object(s) were processed SUCCESSFULLY:

Modify SUCCEEDED: [person] dn1172-ripe   de nis

~~~~~~
The following paragraph(s) do not look like objects
and were NOT PROCESSED:

This is a private email.
It is intended only for the named recipient.

~~~~~~

There are many reasons why the operation may fail for the object, including syntax error, authorisation failure and failed business rules check.

2.7.2 Notifications

There are a number of attributes that may cause notification messages to be generated:

- "notify:"
- "mnt-nfy:"

- "upd-to:"
- "irt-nfy:"
- "ref-nfy:"

Where there are multiple instances of any notification attribute, all the email values will be sent a notification email. Where the notification attribute is contained in a referenced object, for example in a **mntner**, and there are multiple references, all the referenced objects will be taken to form a list of email address. All these addresses will be sent a notification email.

The "notify:" attribute is an option in all object types and is used when an object is successfully updated. The "notify:" attribute of the old version of the object is used if the object is being modified or deleted. The "notify:" attribute of the new version of the object is used if the object is being created. If the update fails for any reason then no notifications will be sent to any "notify:" email addresses.

The optional "mnt-nfy:" and mandatory "upd-to:" attributes can only be included in **mntner** objects.

When a maintained object is updated successfully a notification message will be sent to email addresses contained in the "mnt-nfy:" attributes of the **mntner** objects.

When an **inet(6)num**, **route(6)** or a **domain** object is created, authentication is required from the parent object. If the parent has a "mnt-lower:" (or "mnt-routes:" or "mnt-domains:") attribute, this is the **mntner** that will need to be authenticated against. Otherwise the parent "mnt-by:" attribute be used.

When an update to a maintained object fails the authentication, the notifications are sent to all the email address contained in the "upd-to:" attributes. The rules for finding the appropriate "upd-to:" attributes are the same as for the "mnt-nfy:" above.

The optional "irt-nfy:" attribute is only allowed in an **irt** object. This is used when a reference to an **irt** object is added to, or removed from, an **inetnum** or **inet6num** object (by means of "mnt-irt:" attribute).

The optional "ref-nfy:" attribute is only allowed in an **organisation** object. This is used when a reference to an **organisation** object is added to an object (by means of "org:" attribute).

The format of a notification message is similar to the ACK message. The first section explains why you are being sent this notification. The next section has the email header or IP address details showing where the update came from. The final section shows the changes that were contained in the update message, if it was successful. For a modification a 'diff' is included to show the difference between the original object and the new object. If the update failed for authorisation reasons, it shows the object for which a change was attempted, but not the actual change details.

Each notification message is only sent to a single email address. There is no CC: included in any notification. So when multiple email addresses need to be notified of the same update, each address will receive it's own email. This email will contain all

the notification details of objects from an update message that relate to that email address. For an update with multiple objects, referencing several **mntner** objects, where several objects have a variety of notification attributes, the software builds a matrix of email addresses and updated objects. This ensures that only the appropriate details are sent to each email address.

2.8 Protecting Data

The RIPE Database provides mechanisms to protect objects and control who can make changes to them. In some cases there are also restrictions over who can create, modify or delete certain objects.

- Authentication is the way we determine whose authentication token is attempting to make a change.
- Authorisation is how we decide whether a transaction passing a specific authentication check is allowed to perform a given operation.

Different types of objects in the database require different levels of protection. The server supports multiple authentication methods. Because of the model used, it is not possible to identify who is making an update. The **mntner** objects only hold tokens – for example, encrypted password hashes or references to cryptographic keys. There is no connection between these tokens and any identifiable person.

In order to make the password tokens a stronger form of protection, the encrypted hashes are hidden from public view. These can only be seen if you can supply the clear text password.

2.8.1 Authorisation Model

The **mntner** objects serve as anonymous containers to hold authentication tokens. A reference to a **mntner** object within any object defines authorisation necessary to perform operations on that object or on a set of related objects. Such reference is provided by means of the "mnt-by:", "mnt-lower:", "mnt-routes:", "mnt-domains:", "mnt-ref:" and "mbrs-by-ref:" attributes.

The **mntner** object contains one or more mandatory "auth:" attributes. Each begins with a keyword identifying the authentication method followed by the authentication information or token needed to enforce that method. The **irt** object also has mandatory “auth:” attributes used for authorisation.

When submitting an update that requires authorisation, the authentication information valid for one of the authentication tokens of one of the relevant **mntner** objects should be supplied. Different methods require different authentication information, as shown below.

Authentication methods currently supported include the following:

Method	Description
MD5-PW	This scheme is based on the MD5 hash algorithm. The authentication information stored in the database is a passphrase encrypted using md5-crypt algorithm, which is a concatenation of the "1" string, the salt, and the 128-bit hash output. Because it uses an 8-character salt and an almost unlimited pass phrase and the encrypted hash is hidden from public view, this scheme is quite stable against dictionary attacks. Authentication information is supplied using a "password:" pseudo-attribute. The value of this attribute is a clear-text pass phrase. It can appear anywhere in the body of the message, but not within mail headers. Line continuation is not allowed for this attribute. The attribute and the passphrase should fit on one line. If you split the passphrase across multiple lines this will be treated as a syntax error. Example: auth: MD5 1abcd4321$HyM/GVhPqXkkIMVerxxQ3z
PGPKEY	This is a strong form of authentication. The authentication information is a signature identity pointing to a public key certificate, which is stored in a separate **key-cert** object. The user is authenticated if the transaction is signed by the corresponding private key. The RIPE NCC does not guarantee that a key belongs to any specific entity. Anyone can supply any public keys with any ownership information to the RIPE Database. These keys can be used to protect other objects by checking that the update comes from a user who knows the corresponding secret key. The database software does not check expiry dates of the key or dates when an update message was signed. Example: auth: PGPKEY-1380K9U1
X.509	This is another strong form of authentication. It works in the same way as PGPKEY, but uses an X.509 certificate as the key. This is currently not supported with webupdates or syncupdates. Example: auth: X509-1
SSO	This scheme is based on the RIPE NCC Access single sign-on (SSO) system. It takes the management of these authentication tokens outside of the RIPE Database. To use this, you must first create an account with RIPE NCC Access from the sign-in page: https://access.ripe.net/

	The SSO system was introduced so that when somebody signs in once with RIPE NCC Access, that account authorises them to use certain services that support it, such as Webupdates or Syncupdates. The credential in the **mntner** object uses the keyword SSO followed by the email address used to sign in to your SSO account. You can add many different SSO credentials to a **mntner** object and add your SSO credential to as many **mntner** objects as you wish (providing you have authority to update each **mntner** object using existing authorisation). If you change your email address in your RIPE NCC Access preferences, this will immediately be reflected in any **mntner** objects where this access account is referenced. Authentication using SSO can be done from Webupdates and Syncupdates. You can sign into your RIPE NCC Access account directly from Webupdates and Syncupdates. Using any of the update features from these pages you can create, modify and delete objects. No password is needed - you are already authenticated to make these updates, assuming the object(s) are maintained by one of your SSO mntner objects. Example: auth: SSO dbtest@ripe.net

2.8.2 Protection of Individual Objects

Individual objects can be protected with a **mntner** object. The **mntner** is referenced by the "mnt-by:" attribute in the object. The attribute type is multiple, so several **mntner** objects can be used to protect one object.

Only those **mntner** objects referenced by the "mnt-by:" attributes are authorised to modify or delete the object. Note that authentication checks are logically OR-ed (e.g. A or B or C). If the information required by at least one authentication token from one **mntner** object is supplied, the operation will be authorised. That means that an object's protection level is determined by the weakest authentication method used in the **mntner** objects referenced by that object.

When the "mnt-by:" attribute is added to an object for the first time (as part of object creation or modification), the operation should pass authentication checks for at least one of the **mntner** objects referenced by one of the "mnt-by:" attributes.

If the operation is a modification and the old object already has one or more "mnt-by:" attributes, then one of the **mntner** objects referenced in one of the "mnt-by:" attributes in the old object must authenticate the change. If the old object does not have any "mnt-by:" attributes, then one of the **mntner** objects referenced in one of the "mnt-by:" attributes in the new object must authenticate the change. All new objects must have at least one "mnt-by:". There are still some old **person** and **role** objects that do not have any.

2.8.3 Protection of person and role Objects

When "mnt-by:" was made mandatory on these objects a circular dependency was created. A **person** object must be maintained and a **mntner** must reference an existing **person**. A new user who has no data in the RIPE Database must follow the procedure in 2.3 to get started.

There is a legacy of **person** and **role** object that still do not have a "mnt-by:" attribute. This must be added when the object is next modified.

In order to remind users to maintain their personal data, Warning messages are generated in the acknowledgement message. Every time a **person** or **role** object is referenced that is not maintained, a Warning will remind the user to protect their personal object. Every time a **mntner** object is referenced where the **mntner** has a reference to a **person** or **role** object that is not maintained another Warning message is generated.

2.8.4 Protection of aut-num Object Space

Protection of **aut-num** object space is done using **as-block** objects. The **mntner** object that authorises the creation of **aut-num** objects is specified by any one of the "mnt-lower:" attributes of the parent **as-block** object. When no "mnt-lower:" attribute exists, the **mntner** object from any one of the parent "mnt-by:" attributes is used.

This parent authorisation is only required when an object is created. It is in addition to the authorisation of the individual object itself.

2.8.5 Protection of Address Space

The **inetnum** and **inet6num** objects represent address space allocations and assignments. The "mnt-lower:" attribute is used to reference a **mntner** object that authorises the creation of more specific **inetnum** or **inet6num** objects. If no "mnt-lower:" attribute is specified, one of the "mnt-by:" attributes of the parent object must be used instead.

This parent authorisation is only required when an object is created. It is in addition to the authorisation of the individual object itself.

2.8.6 Protection of Route Object Space

The **route** object creation must satisfy several authentication criteria. This is described in a flow chart [29]

The same sequence applies to **route6** and **inet6num** objects.

2.8.6.1 Pending authentication

In order to simplify the authentication of **route** and **route6**, it's now possible to partially authenticate an update. If a **route** object contains a correct **autnum** authentication, the update will be saved internally until the same update is submitted with correct address space authentication. Some rules apply:

- Submitting a second update with the same hierarchical authorisation as the first update will result in an error since it doesn't complete the updating of the object
- If the update contains any errors other than missing hierarchical authorisation it will fail
- A pending update will only be stored for seven days

When the first update has been stored, notifications are sent to the remaining parties that need to authorise the object. If the remaining parties fail to complete the update within seven days, all parties will be sent a notification that the update failed.

2.8.7 Protection of Objects with Hierarchical Names

RPSL set objects do not have a natural hierarchy of their own but allow hierarchical names, such as the **as-set** object. An **as-set** object may have a non-hierarchical name such as "AS-Foo" or a hierarchical name in the form "AS1:AS-BAR".

For set objects with non-hierarchical names, authorisation corresponds to the rules for individual objects described in Section 2.8.2, 'Protection of Individual Objects'.

If hierarchical names are used, then the creation of an object must be authorised by the object whose primary key is named by everything to the left of the rightmost colon in the primary key name of the object being created. Continuing the hierarchy from above to create the object "AS1:AS-BAR:AS2". This would need to be authorised by "AS1:AS-BAR". This object is considered to be the 'parent' of the object being created.

Hierarchical authorisation is determined by first using any **mntner** object referenced by the parent object's "mnt-lower:" attributes. If none exist, then any **mntner** object referenced by the parent object's "mnt-by:" attributes.

The parent authorisation is required in addition to the authorisation of the individual object itself and only required when an object is created.

2.8.8 Protection of Domain Object Space

Protection of the reverse domain object space for "in-addr.arpa" and ""ip6.arpa"" domains is done with separate methods for creation, deletion and modification. The **domain** object creation is described in a flow chart [30]

This is different to the authentication processes for all other types of objects. Normally when an appropriate object is found to check authorisation against (for example an **inetnum**) the search sequence ends. For other object types, the authorisation will pass or fail with the selected object's referenced maintainers. In

this case, if the authorisation is checked against a selected **inetnum** object and fails, the search sequence continues to look for the parent **domain** object.

For modification and deletion, any **mntner** referenced in a "mnt-by" attribute of the object can authorise the update. Where a deletion fails, **mntners** referenced in a "mnt-domains", "mnt-lower" or "mnt-by" attribute of the corresponding **inet(6)num** object can authorise the deletion. They will be checked in this order. The first attribute type found will be taken as the only one to use.

2.8.9 Protecting Membership of a Set

When membership of a set is specified through the use of the "member-of:" attribute, the server checks the validity of the membership when creating or modifying an object-member. This "member-of:" attribute can be used in **route(6)**, **aut-num** and **inet-rtr** objects. The value of the "member-of:" attribute identifies a **set** object that this object wants to be a member of.

The **set** object must also have a related "mbrs-by-ref:" attribute listing the maintainer of the object wanting to be a member of the **set**. The membership claim of an object with a "member-of:" attribute must be validated. It does that by matching a "mnt-by:" attribute of the object with one of the "mbrs-by-ref:" attributes of the **set** object. If the claim is not valid at the time of creation or modification of an object-member (**route(6)**, **aut-num** or **inet-rtr**), the operation fails. If a **set** object has no "mbrs-by-ref:" attributes, the **set** is defined explicitly by the "members:" attributes in the **set** object. In this case, no other object can validate a claim to be a member of this **set**.

2.8.10 Referencing an irt Object

The **irt** object can be referenced in **inetnum** and **inet6num** objects. This reference is made by adding an optional "mnt-irt:" attribute to the **inet(6)num** object with the name of the **irt** object. Adding a reference to an **irt** object requires authorisation from the **irt** object. Authorisation can be approved by any of the credentials referenced in any of the "auth:" attributes of the **irt** object. The authorisation does not default to the "mnt-by:" attributes of the **irt** object if no suitable credential is found in the "auth:" attributes

Authorisation from the **irt** object is only required when a "mnt-irt:" attribute is added to a referencing object, either on creation or by modification. The **irt** authorisation is required in addition to the authorisation of the individual object itself.

2.8.11 Referencing an Organisation Object

The **organisation** object can be referenced in any object. This reference is made by adding an optional "org:" attribute to the referencing object, along with the name of the **organisation** object. Adding this reference requires authorisation from the **organisation** object. Authorisation can be approved by any of the **mntner** objects referenced in any of the "mnt-ref:" attributes of the **organisation** object. The authorisation does not default to the "mnt-by:" attributes if no suitable maintainer is found in the "mnt-ref:" attributes.

Authorisation from the **organisation** object is only required when an "org:" attribute is added to a referencing object, either on creation or by modification. This is in addition to the authorisation of the referencing object itself.

2.8.12 Reclaim Functionality

There is a process for resource holders to take back or regain control of a resource and any related operational objects. Only the delete operation is possible. The reclaimed objects are deleted by overriding the authentication on those objects. There are very strict rules about which objects can be reclaimed and whose authentication is allowed to override the object's authentication.

For more details, see the article on RIPE Labs [31].

3. Using the RIPE Database Efficiently

Most of this manual provides details of how the RIPE Database works. This section offers advice on how to use the RIPE Database efficiently

3.1 Using the Role Object

The **person** and **role** objects are often said to be interchangeable:

- They share the same name space in the RIPE Ddatabase
- The nic-hdls are only unique across the two object types combined
- A **role** object can be used everywhere that a **person** object can be used.

But these two objects have very different functions. A **person** object holds personal details about an individual. A **role** object should describe a business function or operational unit and may reference the individual people responsible for this activity.
Using **role** objects makes large-scale changes easy. The principle is the same if you have ten objects or 10,000 objects in the database. However, problems most commonly occur when dealing with a very large number of objects.

Many organisations create a large number of objects that directly reference a **person** object, and find themselves in trouble if this person leaves the company. The organisation may be responsible for many objects of different types, possibly with

several different **mntner** objects protecting them, and finding them and getting all the authorisations right to change the references can easily become a problem.

A few basic principles will help to avoid this situation. Only use a **person** object as a holder of personal information, and only reference a **person** object in **role** objects. Reference the **role** objects in all the other objects where contact data is required. If the person responsible for a role changes, then it is simply necessary to modify a few **role** objects to reference a different **person** object. All references to the **role** objects remain valid.

Even if you have only a handful of objects in the database, it is good practice to do this. Your business may grow, and human nature means you will not go back and change things until you have to do so. This is how these objects were intended to be used, but as this practice was never enforced, much of the database still makes direct references to **person** objects.

3.2 Using the Organisation Object

The **organisation** object was introduced long after the RIPE Database was designed. When this object was introduced it was intended to be used in a certain way, but again, this practice was never enforced, and many database users have not adopted it. It is worth knowing how the **organisation** object can make life easier in some situations.
Consider all users as entities, whether they are multinational companies, universities or individuals. To use the RIPE Database, each of these entities needs a set of data objects that represent their business model. They will need:

- People who can be contacted
- Who have defined roles in the business
- Which are responsible for Internet resources
- That need authentication tokens to protect them
- Which may need public keys

This set of objects represents the organisation of the entity. When the entity is an individual who has been assigned some PI space, they may need several objects in the database. Multinational companies may have many hundreds of thousands of objects.
The **organisation** object was introduced as a way of keeping track of these sets of objects. The idea is to put the organisational identity of the entity at the centre by defining its **organisation** object. The organisation's business model can then be mapped out by creating the objects from the list above as appropriate. Each of these objects can be directly 'tagged' with a reference to the **organisation** object using the "org:" attribute. Or for a simplified model, tag the **mntner** objects using the "org:" attribute in the **mntner** object. If all objects are maintained, then there is an indirect reference back to the **organisation** object through the **mntner** objects.

Some multinational companies may have a devolved business model with different parts of the organisation responsible for different parts of their network. In this situation additional **organisation** objects can be created. These objects can reference the main **organisation** object through their own "org:" attribute. This allows users to keep track of the entire company's data or the parts delegated to different sections of the company.
Any bulk changes to data are very much simplified. Tools can be written and deployed more easily. New ideas can be rolled out quickly across an entire data set.

3.3 Abuse Handling

The **irt** (Internet Response Team) object was introduced to identify teams for handling serious network problems like DOS attacks. It should not be used to handle general abuse complaints.

General abuse is handled by the **organisation** object. This should reference an abuse handling **role** object with an “abuse-c:” attribute. This **role** object must include an "abuse-mailbox:" attribute. All address space, represented by **inet(6)num** objects, should reference an **organisation** object either directly or via its less specific objects. This reference defines the abuse handler for this address space and all the more specific address space to that specified by the **inet(6)num** object which references the **organisation** object.

There is a query flag ("-c") which will return the **irt** object, if one exists, for any specified **inet(6)num** object. There is also another query flag ("-b") that will find the indirectly referenced **role** object, extract the "abuse-mailbox:" attribute and return brief details including the email address from the **role** object. For details of how these queries work see Section 2.5, 'Abuse Contacts' in the "RIPE Database Query Reference Manual" [18].

Appendices

A1. Object Attributes

Shown below are the syntax definitions of the object attributes that the RIPE Database supports.

The value of an attribute has a type. Some of the most commonly used types are shown in Table A1. Others are explained in the descriptions of the attributes.

Table A1. Commonly used attribute types

Type	Description
<quad>	<xdigit>.){1,4}
<dlabel>	Domain name label as specified in RFC 1034 [7]. The total length should not exceed 63 characters (octets) <alnum>((-|<alnum>)*<alnum>)?
<action>	Please see RFC 2622 [1]
<address-prefix>	An address prefix is represented as an IPv4 address followed by the character slash "/" followed by an integer in the range from 0 to 32. The following are valid address prefixes: 128.9.128.5/32, 128.9.0.0/16, 0.0.0.0/0; and the following address prefixes are invalid: 0/0, 128.9/16 since 0 or 128.9 are not strings containing four integers. <ipv4-address>/<integer>
<address-prefix-range>	An address prefix range is an address prefix followed by an optional range operator. Please see RFC-2622 [1]
<as-expression>	Please see RFC 2622 [1]
<as-number>	An "AS" string followed by an 32 -bit integer AS<integer>
<condition>	Please see RFC 2622 [1]
<domain-name>	Domain name as specified in RFC 1034 [7] without trailing dot ("."). The total length should not exceed 255 characters (octets) <dlabel>(\.<dlabel>)*
<e-mail>	Eemail address specification as defined in RFC 2822 [8],
<filter>	Please see RFC 2622 [1]
<freeform>	A sequence of Latin 1 characters
<inet-rtr-name>	Specifies the name of an inet-rtr object. It is a <domain-name>.
<ipv4-address>	An IPv4 address is represented as a sequence of four integers in the range from 0 to 255 separated by the

	character dot ("."). For example, 128.9.128.5 represents a valid IPv4 address. [0-9]+(\.[0-9]+){3,3}
<ipv6-address>	<quad>(:<quad>){7,7}
<ipv6-address-prefix>	<ipv6-address>/integer (between 0 and 128)
<ipv6-filter>	Please see RPSLng [14]
<irt-name>	Specifies the name of an **irt** object. It is an <object-name> starting with "IRT-" prefix reserved for this object class.
<mntner-name>	<object-name>
<nic-handle>	From two to four characters, optionally followed by up to five digits optionally followed by a source specification of up to nine characters or two-character country code. Source specification and country codes start with "-". (<alpha>{2,4}([1-9]<digit>{0,5})?(-<alpha> ([a-zA-Z0-9_-]{0,7}<alnum>))?)| (AUTO-<digit>+(<alpha>{2,4})?)
<object-name>	Many objects in RPSL have a name. An <object-name> is made up of letters, digits, the character underscore "_", and the character hyphen "-"; the first character of a name must be a letter, and the last character of a name must be a letter or a digit. The following words are reserved by RPSL, and they can not be used as names: any as-any RS-any peeras and or not atomic from to at action accept announce except refine networks into inbound outbound Names starting with certain prefixes are reserved for certain object types. Names starting with "as-" are reserved for **as-set** names. Names starting with "RS" are reserved for **route-set** names. Names starting with "rtrs-" are reserved for **rtr-set** names. Names starting with "fltr-" are reserved for **filter-set** names. Names starting with "prng-" are reserved for **peering-set** names. Names starting with "irt-" are reserved for **irt** object names.
<org-id>	The 'ORG-' string followed by two to four characters,

	followed by up to five digits followed by a source specification. The first digit must not be "0". Source specification starts with "-" followed by source name up to nine-character length.
<organisation-name>	Is a list of at most 12 words, each at most 64 characters in length. Words can contain alphanumeric characters, asterisk, plus and minus signs, forward slash and backslash, dash, quotes, at sign, commas, dots, underscores, ampersands, exclamation marks, colons, semicolons, brackets and square brackets.
<peering>	Please see RFC 2622 [1]
<person-name>	Is a list of at least two words separated by white space. The first and the last word cannot end with dot ("."). A word is made up of letters, digits, the character underscore "_", and the character hyphen "-"; the first character of a name must be a letter, and the last character of a name must be a letter or a digit.
<protocol>	Please see RFC 2622 [1]
<registry-name>	RIPE
<role-name>	Is a list of at most 12 words, each at most 64 characters in length. Words can contain alphanumeric characters, asterisk, plus and minus signs, forward slash and backslash, dash, quotes, at sign, commas, dots, underscores, ampersands, exclamation marks, colons, semicolons, brackets and square brackets.
<router-expression>	Please see RFC 2622 [1] and RPSLng [14]
<telephone-number>	Contact telephone number. Can take one of the forms: '+' <integer-list> '+' <integer-list> "(" <integer-list> ")" <integer-list> '+' <integer-list> ext. <integer list> '+' <integer-list> "(" integer list ")" <integer-list> ext. <integer-list>
<integer>	An integer
<alpha>	Any alphabetical character.

	[A-Za-z]
<alnum>	Any alphabetical or numeric character. [A-Za-z0-9]
list of	A list of words separated by comma (","). Cannot be empty.
Ripe-list of	A list of words separated by white space. Cannot be empty.
<integer-list>	A list of <integer> with white space or dash ("-") as separators.

Descriptions of the attributes are listed below in the following format:

<attribute_name> <attribute_value(type)>
<description>

Abuse-c: < nic-handle >
References a **role** object holding contact details of an abuse role.

Address: <freeform>
Full postal address of a contact.

Admin-c: <nic-handle>
References an on-site administrative contact.

Aggr-bndry: <as-expression>
Defines a set of ASNs, which form the aggregation boundary.

Aggr-mtd: inbound | outbound [<as-expression>]
Specifies how the aggregate is generated. Please see [1] for more information.

Alias: <domain-name>
Specifies a canonical DNS name for the router.

As-block: <as-number> - <as-number>
Specifies the range of ASNs that the **as-block** object represents. Please see [2] for more information.

As-name: <object-name>
A descriptive name associated with an AS.

As-set: <object-name>
Defines the name of the set.

Auth: <auth-scheme> <scheme-info>
Defines an authentication scheme to be used.
<auth-scheme> and <scheme-info> can take the following values:

<auth-scheme>	<scheme-info>	Description
MD5		This scheme is based on the MD5 hash algorithm. The authentication information stored in the database is a pass phrase encrypted using md5-crypt algorithm, which is a concatenation of the "1" string, the salt, and the 128-bit hash output. Because it uses 8-character salt and an almost unlimited pass phrase, this scheme is more stable against dictionary attacks.
PGPKEY-<id>		Strong scheme of authentication. <id> is the PGP key ID to be used for authentication. This string is the same one that is used in the corresponding **key-cert** object's "key-cert:" attribute.
X5509-<id>		Strongest scheme of authentication. <id> is the auto-generated ID of the X509 certificate to be used for authentication. This string is the same one that is used in the corresponding **key-cert** object's "key-cert:" attribute.

author: <nic-handle>
References a poem author.

aut-num: <as-number>
The autonomous system number.

certif: <public-key>
Contains the public key for a PGP key or an X509 certificate. The value of the public key exported from your local key ring in ASCII-armored format or the certificate from your browser. All the lines of the exported key must be included. For PGP, this includes the begin and end markers and the empty line which separates the header from the key body. For X509 certificates, this includes the BEGIN CERTIFICATE and END CERTIFICATE lines.

changed: <email> [<date>]
Specifies who submitted the update, and when the object was updated. The format of the date is YYYYMMDD; dates in the future are not allowed. If the date is not specified, the database software will add

the date when the update was actually processed. This is user-supplied data that is not verified and has no meaning to anyone else.

components: [ATOMIC] [[<filter>] [protocol <protocol> <filter> ...]]

or: [ATOMIC] [[<ipv6-filter>] [protocol <protocol> <ipv6-filter> ...]]

The "components:" attribute defines what component routes are used to form the aggregate.
<Protocol> is a routing protocol name such as BGP4, OSPF or RIP, and <filter> or <ipv6-filter> is a policy expression.
Please refer to RFC 2622 [1] and RPSLng [14] for more information.

country: <country-code>
Identifies the country. <Country-code> must be a valid two-letter ISO 3166 country code.

default: to <peering> [action <action>] [networks <filter>]
Specifies default routing policies. Please refer to RFC 2622 [1] for more information.

descr: <freeform>
A short description related to the object

domain: <reverse-domain-name>
reverse delegation for IPv4 or IPv6 address space.

e-mail: <e-mail>
Specifies an email address of a person, role, organisation or IRT team.

encryption: PGPKEY-<id>
References a **key-cert** object representing a CSIRT public key used to encrypt correspondence sent to the CSIRT. <Id> is the key-id of the PGP public key in eight digit hexadecimal format without "0x" prefix.

export: to <peering-1> [action <action-1>]
. . .
to <peering-N> [action <action-N>]
announce <filter>
Specifies an export policy expression. Please refer to RFC 2622 [1] for more information.

export-comps: <filter> or <ipv6-filter>
Specifies an RPSL filter that matches the more specifics that need to be exported outside the aggregation boundary. Please refer to RFC 2622 [1] and RPSLng [14] for more information.

export-via: [protocol <protocol-1>] [into <protocol-2>]

afi <afi-list>

<peering-1>

to <peering-2> [action <action-1>; <action-2>; ... <action-N>;]

...

<peering-3>

to <peering-M> [action <action-1>; <action-2>; ... <action-N>;]

announce <filter>

Specifies export policy expression for non-adjacent networks. Please refer to draft [33] for more information.

fax-no: <telephone-number>
The fax number of a contact.

filter: <filter>
Defines the set's policy filter, a logical expression which when applied to a set of routes returns a subset of these routes. Please refer to RFC 2622 [1] for more information.

filter-set: <object-name>
Defines the name of the filter. Please refer to RFC 2622 [1] for more information.

fingerpr: <generated>
A fingerprint of a key certificate generated by the database. Please refer to RFC 2726 [9] for detailed description of this attribute.

form: **FORM** <string>
Specifies the identifier of a registered poem object.

holes: list of <address-prefix> or <ipv6-address-prefix>
Lists the component address prefixes that are not reachable through the aggregate route (perhaps that part of the address space is unallocated). Please refer to RFC 2622 [1] and RPSLng [14] for more information.

ifaddr: <ipv4-address> masklen <integer> [action <action>]
Specifies an interface address within an Internet router. Please refer to RFC 2622 [1] for more information.

import: [protocol <protocol-1>] [into <protocol-2>]
from <peering-1> [action <action-1>]
. . .
from <peering-N> [action <action-N>]
accept <filter>
Specifies import policy expression. Please refer to RFC 2622 [1] for more information.

import-via: [protocol <protocol-1>] [into <protocol-2>]

afi <afi-list>

<peering-1>

from <peering-2> [action <action-1>; <action-2>; ... <action-N>;]

...

<peering-3>

from <peering-M> [action <action-1>; <action-2>; ... <action-N>;]

accept (<filter>|<filter> except <importexpression>|

<filter> refine <importexpression>)

Specifies import policy expression for non-adjacent networks. Please refer to [33] for more information.

inetnum: <ipv4-address> - <ipv4-address>
Specifies a range of IPv4 addresses. The ending address should be greater than the starting one.

inet6num: <ipv6-address>/<prefix-length>
Specifies a range of IPv6 addresses in prefix notation. The <prefix length> is an integer in the range from 0 to 128.

inet-rtr: <domain-name>
Fully qualified DNS name of the **inet-rtr** without trailing ".". Please refer to RFC 2622 [1] for more information.

inject: [at <router-expression>]
[action <action>]
[upon <condition>]
Specifies which routers perform the aggregation and when they perform it. In route objects, the router expression can contain only IPv4 expressions, and in route6 object it can only cotain IPv6 expressions. Please refer to RFC 2622 [1] and RPSLng [14] for more information.

interface: <ipv4-address> or <ipv6-address> masklen <masklen> <integer> [action <action>]

[tunnel <remote-endpoint-address>,<encapsulation>]

Specifies a multiprotocol interface address within an Internet router. Please refer to RPSLng [14] for more information.

irt: <irt-name>
A unique identifier of an **irt** object. The name should start with the prefix "IRT-", reserved for this type of object.

irt-nfy: <e-mail>
Specifies the email address to be notified when a reference to the **irt** object is added or removed.

key-cert: PGPKEY-<id>
Defines the public key stored in the database. <Id> is the ID of the PGP public key in 8-digit hexadecimal format without "0x" prefix.

local-as: <as-number>
Specifies the autonomous system that operates the router. Please refer to RFC 2622 [1] for more information.

method: <generated>
Defines the type of the public key. Currently the only methods supported are "PGP" and "X509". Please refer to RFC 2726 [9] for detailed description of this attribute.

member-of: list of <set-name>
This attribute can be used in the **route**, **route6, aut-num** and **inet-rtr** classes. The value of the "member-of:" attribute identifies a set object that this object wants to be a member of. This claim, however, should be acknowledged by a respective "mbrs-by-ref:" attribute in the referenced object. Please refer to RFC 2622 [1] for more information.

members: list of <as-number> *or* <as-set-name>

or

members: list of <address-prefix-range>*or* <route-set-name><range-operator>

or

members: list of <inet-rtr-name> *or* <rtr-set-name> *or* <ipv4 address>

Lists the members of the set. The first form appears in the **as-set** object. The syntax of <as-set-name> is the same as the syntax of <object-name>. The second form appears in the **route-set** object. The syntax of <route-set-name> is the same as the syntax of <object-name>. The third form appears in the **rtr-set** object. The syntax of <intet-rtr-name> is the same as the syntax of <object-name>. Please refer to RFC-2622 [1] for more information.

mbrs-by-ref: list of <mntner-name> | ANY
This attribute can be used in all "set" objects; it allows indirect population of a set. If this attribute is used, the set also includes objects of the corresponding type (**aut-num** objects for **as-set**, for example) that are protected by one of these maintainers and whose "member-of:" attributes refer to the name of the set. If the value of a "mbrs-by-ref:" attribute is ANY, any object of the corresponding type referring to the set is a member of the set. If the "mbrs-by-ref:" attribute is missing, the set is defined explicitly by the "members:" attribute.

mntner: <object-name>
A unique identifier of the mntner object.

mnt-by: list of <mntner-name>
Specifies the identifier of a registered **mntner** object used for

authorisation of operations performed on the object that contains this attribute.

mnt-domains: list of <mntner-name>

Specifies the identifier of a registered **mntner** object used for reverse domain authorisation. Controls creation of **domain** objects. The authentication method of this **mntner** object will be used to authorise the creation of an exact match or more specific reverse **domain** object.

mnt-irt: list of <irt-name>
May appear in an **inetnum** or **inet6num** object. It references an existing **irt** object representing CSIRT that handles security incidents or general abuse handler for the address space specified by the **inetnum** or **inet6num** object.

mnt-lower: list of <mntner-name>
Specifies the identifier of a registered **mntner** object used for hierarchical authorisation. Controls creation of objects one level more specific in the hierarchy of an object type (only for **inetnum**, **inet6num**, **as-block**, **aut-num**, **route, route6** objects). The authentication method of this **mntner** object will then be used to authorise the creation of any object one level more specific to the object that contains the "mnt-lower:" attribute.

mnt-nfy: <e-mail>
Specifies the email address to be notified when an object protected by a **mntner** is successfully updated.

mnt-ref: list of <mntner-name>

Specifies the **mntner** objects that are entitled to add references to the **organisation** object from other objects.

mnt-routes: <mnt-name> [{ list of <address-prefix-range> } | ANY]
May be used in an **aut-num**, **inetnum**, **inet6num**, **route** or **route6** object. Specifies the identifier of a registered **mntner** object that controls the authorisation of the creation of **route** and **route6** objects. After the reference to the maintainer, an optional list of prefix ranges inside of curly braces or the keyword "ANY" may follow. The default, when no additional set items are specified, is "ANY" or all more specifics. The address prefix range can contain only IPv4 prefix ranges in **inetnum** and **route** objects, only IPv6 prefix ranges in **inet6num** and **route6** objects, and it can contain both IPv4 and IPv6 prefix ranges in **aut-num** objects. Please refer to RFC-2622 [1] and RPSLng [14] for more information.

mp-default: to <peering> [action <action>] [networks <filter>]

Specifies default multiprotocol routing policies. Please refer to RPSLng [4] for more information.

mp-export:

```
[protocol <protocol-1>] [into <protocol-1>]
afi <afi-list>
to <peering-1> [action <action-1>]
   .
   .
   .
to <peering-N> [action <action-N>]
announce <filter>
```

Specifies a multiprotocol export policy expression. Please refer to RPSLng [14] for more information.

mp-filter:

Defines the set's multiprotocol policy filter. Please refer to RPSLng [14] for more information.

mp-import: [protocol <protocol-1>] [into <protocol-1>]

```
afi <afi-list>
from <peering-1> [action <action-1>]
   .
   .
   .
from <peering-N> [action <action-N>]
accept (<filter>|<filter> except <importexpression>|
     <filter> refine <importexpression>)
```

Specifies multiprotocol import policy expression. Please refer to RPSLng [14] for more information.

mp-members: afi <afi-list> list of <address-prefix-range> or

<route-set-name> or

<route-set-name><range-operator>

Lists the multiprotocol members of the set. Refer to RPSLng [14] for more information.

mp-peer: <protocol> afi <afi> <ipv4- or ipv6- address> <options>

| <protocol> <inet-rtr-name> <options>

| <protocol> <rtr-set-name> <options>

| <protocol> <peering-set-name> <options>

Specifies the details of any (interior or exterior) multiprotocol router peerings. Please refer to RPSLng [14] for more information.

mp-peering: afi <afi> <peering>

Defines a multiprotocol peering that can be used for importing or exporting routes. Please see RPSLng [14] for more information.

netname: <netname>
Specifies the name of a range of IP address space. The syntax of the <netname> attribute is the same as the syntax of the <object-name> attribute, but it does not have a restriction on RPSL reserved prefixes.

nic-hdl: <nic-handle>
Specifies the NIC handle of a **role** or **person** object. When creating an object, one can also specify an "AUTO" NIC handle by setting the value of the attribute to "AUTO-1" or AUTO-1 <Initials>. In such case the database software will assign the NIC handle automatically.

notify: <e-mail>
Specifies the email address to which notifications of changes to an object should be sent.

nserver: <domain-name> [<ipv4-address> | <ipv6-address>]
Specifies the name servers of the domain optionally followed by a glue record.

org: <org-id>

This optional attribute may be used in any object type. It references an existing **organisation** object representing the entity that holds the resource, (in the cases where the RIPE Database object represents an Internet resource). In other objects, it can be used to specify the business relations. The value of this attribute is the ID of the **organisation** object. It is required in the **inetnum** and **inet6num** objects with ALLOCATED-BY-RIR, ALLOCATED PA, ALLOCATED PI and ALLOCATED UNSPECIFIED status values.

The "org:" attribute is single-valued in the **inetnum**, **inet6num** and **aut-num** objects, and it is multi-valued in all other objects.

org-name: <organisation-name>

Specifies the name of the organisation that this **organisation** object represents in the RIPE Database. This is an ASCII-only text attribute. This restriction is because the attribute is a look-up key and the RIPE Database protocol does not allow specifying character sets in queries. The user can put the name of the organisation using Latin 1 character set in the "descr:" attribute if required. But any use of non-ASCII characters in any object may cause problems during the update process.

org-type:

Specifies the type of the organisation. The possible values are:

- **IANA** for Internet Assigned Numbers Authority
- **RIR** for Regional Internet Registries
- **LIR** for Local Internet Registries
- **WHITEPAGES** for linking people well-known within the industry
- **DIRECT-ASSIGNMENT** for organisations that have direct contracts with the RIPE NCC
- **OTHER** for all other organisations

organisation: <org-id>

Specifies the ID of an organisation object. When creating this object, the value of this attribute is auto generated. The user has to specify an "AUTO" ID by setting the value to "AUTO-1" or "AUTO-1<letterCombination>, so the database will assign the ID automatically.

origin: <as-number>
Specifies the AS that originates the route. The corresponding **aut-num** object should exist in the database.

owner: <generated>
Specifies the owner of the public key. Please refer to RFC 2726 [9] for detailed description of this attribute.

peer:		<protocol><ipv4-address><options>
		<protocol><inet-rtr-name><options>
		<protocol><rtr-set-name><options>
		<protocol><peering-set-name><options>

May appear in an **inet-rtr** object. Specifies a protocol peering with another router. Please refer to RFC 2622 [1] for more information.

peering: <peering>
Defines a peering that can be used for importing or exporting routes. Please refer to RFC 2622 [1] for more information.

peering-set: <object-name>
Specifies the name of the peering-set. Please refer to RFC 2622 [1] for more information.

person: <person-name>
Specifies the full name of an administrative, technical or zone contact person for other objects in the database. <Person name> cannot

contain titles such as "Dr.", "Prof.", "MV", "Ms.", "Mr.", etc. It is composed of alphabetic characters.

peering-set: <object-name>
Specifies the name of the peering-set. Please refer to RFC 2622 [1] for more information.

phone: <telephone-number>
Specifies a telephone number of the contact.

poem: POEM <string>
Specifies the title of a poem.

poetic-form: FORM <string>
Specifies the poem type.

ref-nfy: <e-mail>
Specifies the email address to be notified when a reference to the **organisation** object is added or removed. An email address as defined in RFC 2822 [8].

referral-by: <mntner-name>
This attribute is required in the **mntnr** object. It is not used by the database software.

remarks: <freeform>
Contains remarks.

role: <person-name>
Specifies the full name of a role entity, e.g. RIPE DBM.

route: <address-prefix>
Specifies the prefix of the interAS route. Together with the "origin:" attribute, constitutes a primary key of the **route** object.

route6: <ipv6-address>/<prefix-length>
Specifies an IPv6 prefix. The <prefix length> is an integer in the range from 0 to 128. This is the prefix of the interAS route. Together with the "origin:" attribute, constitutes a primary key of the **route6** object.

route-set: <object-name>
Specifies the name of the route set. It is a primary key for the route-set object. Please refer to RFC 2622 [1] for more information.

rtr-set: <object-name>
Defines the name of the **rtr-set**. Please refer to RFC 2622 [1] for more information.

signature: PGPKEY-<id>
References a **key-cert** object representing a CSIRT public key used by the team to sign their correspondence. <Id> is the key-id of the PGP public key in 8-digit hexadecimal format without "0x" prefix.

source: <registry-name>
Specifies the registry where the object is registered. Should be "RIPE" for the RIPE Database.

status: <status>
Specifies the status of the address range represented by inetnum or **inet6num** object. For an **inetnum** object <status> must have one of these values:

- ALLOCATED PA
- ALLOCATED PI
- ALLOCATED UNSPECIFIED
- ASSIGNED PA
- ASSIGNED PI
- ASSIGNED ANYCAST
- LIR-PARTITIONED PA
- LIR-PARTITIONED PI
- SUB-ALLOCATED PA
- EARLY-REGISTRATION
- NOT-SET

Please refer to the RIPE document "IPv4 Address Allocation and Assignment Policies in the RIPE NCC Service Region" for further information. Please refer to [10] regarding usage of the LIR-PARTITIONED status value.
For **inet6num**, <status> can have one of the following values:

- ALLOCATED-BY-RIR - For allocations made by an RIR to an LIR.
- ALLOCATED-BY-LIR - For allocations made by an LIR or an LIR's downstream customer to another downstream organisation.
- AGGREGATED-BY-LIR - For aggregations of assignments with the same prefix length
- ASSIGNED - For assignments made to End User sites.
- ASSIGNED PI
- ASSIGNED ANYCAST

Please refer to [13] regarding usage of the status value for **inet6num** objects.

tech-c: <nic-handle>
References a technical contact.

text: <freeform>
Contains text of the poem. Must be humourous, but not malicious or insulting.

upd-to: <e-mail>
Specifies the email address to be notified when an object protected by a mntner is unsuccessfully updated. See also Section 3.5.2, 'Notifications'.

zone-c: <nic-handle>
References a zone contact.

A2. Copyright Information

A2.1 RIPE Database Copyright

The information in the RIPE Database is available to the public subject to the RIPE Database Terms and Conditions [25].

Terms and Conditions:

http://www.ripe.net/db/support/db-terms-conditions.pdf

A2.2 RIPE NCC Copyright

© RIPE NCC 2001 - 2013

Acknowledgements

The authors wish to acknowledge the work done by the original developers of version 3.0 of the RIPE Database software and infrastructure at the RIPE NCC:

Andrei Robachevsky, Daniele Arena, Marek Bukowy, Engin Gunduz, Roman Karpiuk, Shane Kerr, Ambrose M.R. Magee, Chris Ottrey and Filippo Portera.

Those who have continued its development include Denis Walker, Katie Petruska, Agoston Horvath, Can Bican, Tiago Anteo, Jos I Boumans, Luis Motta Campos, Erik Broes and Menno Blom, Edward Shryane, Angela Sjoholm, Theodoros Polychniatis

References

[1] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg and M. Terpstra, "Routing Policy Specification Language (RPSL)", RFC 2622, June 1999

[2] C. Villamizar, C. Alaettinoglu, D. Meyer and S. Murphy, "Routing Policy System Security", RFC 2725, December 1999

[3] D. Meyer, J. Schmitz, C. Orange, M. Prior and C. Alaettinoglu, "Using RPSL in Practice", RFC 2650, August 1999

[4] T. Bates, E. Gerich, L. Joncheray, J.M. Jouanigot, D. Karrenberg, M. Terpstra and J. Yu, "Representation of IP Routing Policies in a Routing Registry", ripe-181, October 1994

[5] RIPE Database User Manual - Getting Started

[7] P. Mockapetris, "Domain names - Concepts and Facilities", RFC 1034, November 1987

[8] P. Resnick, ed., "Internet Message Format", RFC 2822, April 2001

[9] J. Zsako, "PGP Authentication for RIPE Database Updates", RFC 2726, December 1999

[10]) N. Nimpuno and A.Robachevsky, "New Value of the "status:" Attribute for Inetnum Objects (LIR-PARTITIONED)", ripe-239, June 2002

[13] J.S.L. Damas and L. Vegoda, "New Values of the "status:" Attribute for **inet6num** Objects", ripe-243, August 2002

[14] L. Blunk, J. Damas, F. Parent and A. Robachevsky, Routing Policy Specification Language next generation (RPSLng), August 2004

[16] RIPE NCC service region, http://www.ripe.net/membership/maps/index.html

[18] RIPE Database Query Reference Manual

[19] RIPE document store, http://www.ripe.net/ripe/docs/

[20] Protocol details for syncupdates, see http://www.ripe.net/data-tools/db/syncupdates/manual

[21] Sample clients for syncupdates,

https://www.ripe.net/data-tools/db/syncupdates/syncupdate-perl-script

https://apps.db.ripe.net/syncupdates/simple-rpsl.html

[22] Webupdates https://apps.db.ripe.net/webupdates/

[23] J. Callas, L. Donnerhacke, H. Finney and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998

[25] RipeIPE RIPE Database Terms and Conditions, http://www.ripe.net/db/support/db-terms-conditions.pdf

[26] New organisation startup https://apps.db.ripe.net/startup/index.html

[27] White Pages Instructions http://www.ripe.net/db/support/white-pages-instructions.pdf

[28] Regional Internet Registries (RIR), http://www.ripe.net/info/resource-admin/index.html

[29] Route object creation authorisation flow chart
http://www.ripe.net/data-tools/db/faq/faq-route-object/what-are-the-authorisation-rules-for-route-object-creation

[30] Domain object creation authorisation chart
http://www.ripe.net/data-tools/dns/reverse-dns/create.pdf

[31] Reclaim functionality
https://labs.ripe.net/Members/denis/reclaim-functionality-for-resource-holders
[32] Dry-run feature
https://labs.ripe.net/Members/denis/dry-run-testing-in-the-ripe-database
[33] J. Snijders, "The 'import-via' and 'export-via' attributes in RPSL Policy Specifications" (draft), http://tools.ietf.org/html/draft-snijders-rpsl-via-02, August 2013.