



ENUM registrar manual

version: 1.0
date: 16-02-2009
(c) 2009 ENUM NL

Table of Contents

1 Role Model.....	5
1.1 Registry.....	5
1.2 Registrar.....	5
1.3 Validation Entity (VE).....	5
1.4 Registrant (phone number user).....	5
1.5 Number holder.....	6
2 Administrative procedures.....	6
2.1 Registrar signup.....	6
2.2 Validation entity signup.....	6
3 Supported number ranges.....	6
3.1 Geographic numbers.....	6
3.1.1 Number definition and usage.....	6
3.1.2 ENUM registration guidelines.....	7
3.1.3 Examples.....	7
3.2 Mobile numbers.....	7
3.2.1 Number definition and usage.....	7
3.2.2 ENUM registration guidelines.....	7
3.2.3 Example.....	7
3.3 Information numbers.....	7
3.3.1 Definition and usage.....	7
3.3.2 ENUM delegation guidelines.....	8
3.3.3 Example.....	8
3.4 Business numbers.....	8
3.4.1 Definition and usage.....	8
3.4.2 ENUM delegation guidelines.....	8
3.4.3 Examples.....	8
3.5 Location independent fixed line numbers.....	8
3.5.1 Definition and usage.....	8
3.5.2 ENUM delegation guidelines.....	8
3.5.3 Example.....	9
4 Extensible Provisioning Protocol EPP.....	9
4.1 EPP overview.....	9
4.1.1 Registry production server.....	9
4.1.2 Registry test server.....	10
5 Registry object types.....	10
5.1 Object relation.....	10
5.2 Object ownership and access.....	11
5.3 Object types.....	11
5.3.1 Number type object.....	11
5.3.2 Contact type object.....	13
5.3.3 Nameserver type object.....	15
5.3.4 Token type object.....	17
6 EPP commands and messages.....	19
6.1 Transactions (transform commands).....	19
6.1.1 create number.....	19
6.1.2 update number.....	21
6.1.3 delete number.....	23
6.1.4 renew number (supply a new token).....	24
6.1.5 create contact.....	26

6.1.6 update contact.....	28
6.1.7 delete contact.....	30
6.1.8 Create nameserverset.....	31
6.1.9 Update nameserverset.....	32
6.1.10 Delete nameserverset.....	34
6.1.11 transfer number.....	35
6.2 Query commands.....	38
6.2.1 „info“ command.....	38
6.2.2 „check“ command.....	39
6.2.3 „hello“ command.....	40
6.2.4 „poll“ command.....	40
6.3 EPP response extension.....	41
6.3.1 condition description.....	41
6.3.2 response example.....	42
6.4 EPP message queue.....	42
6.4.1 Message format.....	42
6.4.2 Queue policy.....	43
6.4.3 Message type definitions.....	43
7 Example Transactions.....	45
7.1 EPP login.....	45
7.2 Initial object creation.....	45
7.3 ENUM Domain for a geographic number.....	45
8 Test Cases.....	46
8.1 Contacts.....	46
8.1.1 Create a new contact.....	46
8.1.2 Update attributes of an existing contact.....	47
8.1.3 Delete a contact from the database.....	47
8.2 Nameserversets.....	47
8.2.1 Create a new nameserverset.....	47
8.2.2 Update attributes of an existing nameserverset.....	47
8.2.3 Delete a nameserverset from the database.....	48
8.3 Numbers.....	48
8.3.1 Create number: new ENUM number delegation.....	48
8.3.2 Update number: change delegation attributes.....	48
8.3.3 Renew number: supply a new token for a delegation.....	49
8.3.4 Transfer number: move the delegation from one registrar to another registrar.....	49
8.3.5 Delete number: remove the delegation from the database.....	49
8.4 General.....	49
8.4.1 Info: ask for information on an object.....	49
8.4.2 Check whether a number is already in the database.....	50
8.4.3 Poll for new messages for the registrar.....	50
9 Validation.....	51
9.1 The validation token.....	51
9.1.1 Validation token attributes.....	51
9.1.2 Token signature.....	51
9.2 Generic token verification process.....	52
10 Software toolkit.....	53
10.1 Installation.....	53
10.2 Example installation details.....	54
10.2.1 Testing - create a private key and certificate.....	54
10.2.2 Testing - create a token.....	54
10.2.3 Sample-directory.....	55

1 Role Model

Several parties act together to organize ENUM domain delegations. This overview explains the individual roles and responsibilities. Please note that while we describe here each role as a separate entity, that need not be the case in practice.

RFC 4725 (<http://www.ietf.org/rfc/rfc4725.txt>) gives additional information about the ENUM registration role model, especially about the validation architecture.

1.1 Registry

The registry is the single entity which operates the master database of delegated ENUM domains within country and runs the authoritative nameservers for the relevant zone under e164.arpa. In The Netherlands, this role has been contracted to **ENUM NL**.

The sole purpose of the registry is to delegate domains. It will not offer any other ENUM-related services like e.g. a SIP-community, a VoIP Gateway, or NAPTR record hosting. It will only populate the 1.3.e164.arpa zone with NS records delegating the ENUM domain for individual numbers to other nameservers.

1.2 Registrar

A registrar requests ENUM domain delegations at the registry on behalf of a telephone number user. This is the same role registrars fulfill in the ccTLD and gTLD world. Direct registration from a number user is not permitted. All delegation requests must be channeled through a registrar.

1.3 Validation Entity (VE)

The validation entity is the party which confirms that the requesting telephone number user is indeed authorized to register the ENUM domain in question. This is the role which needs to solve the „Validation Problem“: Making sure that the control over the ENUM domain always follows the right-to-use of the corresponding E.164 telephone number.

The VE creates tokens (signed XML documents) to be used by a registrar to register ENUM domains at the registry which checks these certificates. These tokens document that the VE has checked whether a prospective ENUM domain owner has the right to use for the corresponding telephone number.

The registrar contract with the registry states that the registrar is responsible for solving the validation problem. The role of the VE and the Validation Tokens have been introduced to allow an outsourcing of that functionality to an external party.

Validation Tokens link the following information together:

1. E.164 Number
2. Registrar
3. Validation entity
4. Validation method
5. Validity timeframe
6. Number user Identity (optional)

Please note that such a token gives the registrar control over the ENUM domain of the number user. As the number user never interacts directly with the registry, such a token is the one and only certificate that the registrar does indeed work on behalf of the number user.

1.4 Registrant (phone number user)

The registrant is the end-user in the whole setup. He is the person (or organization) who is actually using the telephone number for which ENUM services are desired. This ownership of a number can come

either through a direct assignment of the number from the number authority OPTA (business numbers and information numbers), or via assignment by a telephony service provider.

1.5 Number holder

The rightful user of a telephone number is internationally referred to as "number holder". In The Netherlands a number holder more generally refers to a company that has been assigned telephone numbers from the number authority for further distribution among their customers. (The company that is in control of the telephony services associated with the number.)

Where applied in this manual the term "number holder" is restricted to the meaning of "rightful user of the phone number in question".

2 Administrative procedures

2.1 Registrar signup

Signing up with the registry for service involves the following steps:

- Fulfill the criteria as described in the registrar contract between registrar and ENUM NL
- Sign the registrar contract, receive registry account data from ENUM NL
- Set up communication with the registry

To be able to provision ENUM domains with the registry, a registrar will require services from a validation entity. Those services can either be provided by the registrar itself (so a single party is acting as registrar plus validation entity) or they can be acquired by a third party (external validation entity).

2.2 Validation entity signup

Signing up with the the registry as Validation Entity involves the following steps:

- Sign the Validation Entity contract between the VE and ENUM NL
- Submit intended validation methods, receive confirmation to use these methods from ENUM NL
- Provide the registry with certificates to be used to verify validation tokens

After those steps have been completed, the validation entity may produce tokens which can be used to provision ENUM domains.

3 Supported number ranges

The Dutch „Nummerplan telefoon- en ISDN-diensten“ as released by the Ministry of Economic Affairs on <http://www.ez.nl> is at the basis for the following section.

Terminology

- **NDC:** „national destination code“ (sometimes referred to as „area code“)

3.1 Geographic numbers

3.1.1 Number definition and usage

Geographic numbers are numbers with a specific geographical destination. Those numbers consist of a fixed length of 9 digits (excluding the starting "0" of the local prefix), with a variable length national

destination code (length between 2 and 3 digits), and a variable length subscriber number (length between 6 and 7 digits).

National destination codes for geographic numbers currently start with digits 1 – 5 and 7, but not all NDC starting with those digits indicate geographic numbers.

Geographic numbers are traditionally used to address fixed location PSTN/ISDN lines in home/office/enterprise environments.

3.1.2 ENUM registration guidelines

Registration of ENUM domains must only be done for complete numbers (including the subscribers number). Number block delegations are allowed, but only if the whole number block is assigned to one subscriber.

When provisioning number blocks, extensions must be excluded from the registration, and have to be managed on the registrar level in the DNS.

3.1.3 Examples

- **Original Number:** „+31 20 1234567“, where „+31“ indicates the country code for The Netherlands, „20“ indicates a geographic number in Amsterdam and „1234567“ indicates the subscriber number.
- **Number to be provisioned:** „+31201234567“
- **Original Number:** „+31 30 2233400“ (with usage of the whole number block by one subscriber), where „+31“ indicates the country code for The Netherlands, „30“ indicates a geographic number in Utrecht and „2233400“ indicates the first subscriber number within a block size 100 (numbers 2233400 - 2233499).
- **Number to be provisioned:** „+313022334“, since extensions have to be excluded.

3.2 Mobile numbers

3.2.1 Number definition and usage

Mobile numbers are commonly used to address mobile phones, SIM cards and related equipment. Numbers start with 1-digit NDC “6” and a fixed length subscriber number of 8 digits.

3.2.2 ENUM registration guidelines

The full number as assigned to the subscriber must be used to acquire the corresponding ENUM domain. Block delegations must not be performed, neither in case of blocks assigned to mobile networks nor in the case of number block allocations to eg enterprises, since both cases would prevent subscribers to choose a different registrar for a certain number from within that block.

3.2.3 Example

- **Original number:** „+31 6 12345678“, where „+31“ indicates the country code for The Netherlands, „6“ indicates the national destination code, and „12345678“ indicates the subscriber number.
- **Number to be provisioned:** „+31612345678“

3.3 Information numbers

3.3.1 Definition and usage

These numbers are used for information-services, both as a free service (0800) or as a commercial service with a variable fee (09xx).

Numbers of this type start with 3-digit NDC “800”, “900”, “906” or “909”, and a fixed-length subscriber number of 4 or 7 digits.

3.3.2 ENUM delegation guidelines

The full number as assigned to the subscriber must be used to acquire the corresponding ENUM domain. Block delegations must not be performed, neither in case of blocks assigned to call handling services nor in the case of number block allocations to eg enterprises, since both cases would prevent subscribers to choose a different registrar for a certain number from within that block.

3.3.3 Example

- **Original number:** „+31 800 1234“, where „+31“ indicates the country code for The Netherlands, „800“ indicates the national destination code, and „1234“ indicates the subscriber number.
- **Number to be provisioned:** „+318001234“

3.4 Business numbers

3.4.1 Definition and usage

Business numbers are assigned directly to enterprises by OPTA (the Dutch regulator for post and electronic communications). This provides enterprises with a uniform number (range) for geographically different locations.

Numbers of this type start with NDC “88” followed by a fixed-length subscriber number of 7 digits.

3.4.2 ENUM delegation guidelines

Registration of ENUM domains must only be done for complete numbers (including the subscribers number). Number block delegations are allowed, but only if the whole number block is assigned to one subscriber.

When provisioning number blocks, extensions must be excluded from the registration, and have to be managed on the registrar level in the DNS.

3.4.3 Examples

- **Original Number:** „+31 88 1234567“, where „+31“ indicates the country code for The Netherlands, „88“ indicates the national destination code and „1234567“ indicates the subscriber number.
- **Number to be provisioned:** „+31881234567“

- **Original Number:** „+31 88 2233400“ (with usage of the whole number block by one subscriber), where „+31“ indicates the country code for The Netherlands, „88“ indicates the national destination code and „2233400“ indicates the first subscriber number within a block size 100 (numbers 2233400 - 2233499).
- **Number to be provisioned:** „+318822334“, since extensions have to be excluded.

3.5 Location independent fixed line numbers

3.5.1 Definition and usage

Location independent fixed line numbers start with 3-digit NDC “84”, “85”, “87” or “91”, followed by a fixed-length subscriber number of 7 digits.

Numbers in this range are not yet heavily used, but due to the fact that such numbers are portable nationwide usage for home telephony services based on PSTN or IP is expected.

3.5.2 ENUM delegation guidelines

Delegations have to be performed on the subscriber number level, since delegation on block level would prevent single subscribers to transfer their number to a different registrar.

3.5.3 Example

- **Original Number:** „+31 85 1234567“, where „+31“ indicates the country code for The Netherlands, „85“ indicates the national destination code and „1234567“ indicates the subscriber number.
- **Number to be provisioned:** „+31851234567“

4 Extensible Provisioning Protocol EPP

All transactions (except the simple existence check via finger, and some manual signup operations) use the EPP interface of the registry. EPP is a provisioning protocol developed by the IETF for the gTLD and ccTLD world. The RFC defining the protocol has been released in 2004. Some registries have already adopted EPP, most with slight changes to accommodate local requirements.

An early evaluation by enum.at showed that the predefined object within vanilla EPP are not sufficient to handle the needs of an ENUM registry. To keep the protocol clean they opted to use the predefined EPP operations but use objects specifically designed for ENUM.

4.1 EPP overview

EPP is an XML-based connection-oriented protocol. On the transport side it uses a TLS-encrypted TCP connection to exchange XML documents (with minimal framing according to [RFC3730](#)) between server and client.

4.1.1 Registry production server

The server listens on TCP port **700** on host **ers.enum.nl** for incoming connections. After an initial TLS handshake, during which the client can verify the identity of the server by its certificate, the server will send a greeting message announcing the supported EPP version:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <greeting>
    <svID> enum-nl test </svID>
    <svDate>2004-10-18T10:57:18.31Z</svDate>
    <svcMenu>
      <version>1.0</version>
      <lang>en</lang>
      <lang>de</lang>
      <objURI>http://rxd.enum.nl/enum-number-1.0</objURI>
      <objURI>http://rxd.enum.nl/enum-contact-1.0</objURI>
      <objURI>http://rxd.enum.nl/enum-nsset-1.0</objURI>
      <objURI>http://rxd.enum.nl/enum-token-1.0</objURI>
    </svcMenu>
    <dcp>
      <access><all/></access>
      <statement>
        <purpose><admin/><prov/></purpose>
        <recipient><ours/><public/></recipient>
        <retention><stated/></retention>
      </statement>
    </dcp>
  </greeting>
</epp>
```

Next the client must login into the server. To do that, it needs to send a <login> XML frame.

Once the connection is established, all further transaction will consist of an XML frame sent by the client which contains a command to the server. The server will process the request and answer back with a response XML frame. Most transactions will be synchronous: the changes to the master database

of the registry will be done before the answer is sent back to the client. Please be aware that changes to the 1.3.e164.arpa zone will not happen in realtime. The updates will be reflected after the periodical update of the name servers. Due to the caching behaviour of the DNS, full propagation may take up to 48 hours.

4.1.2 Registry test server

An EPP test server will be available at **eto.enum.nl**, port **700**.

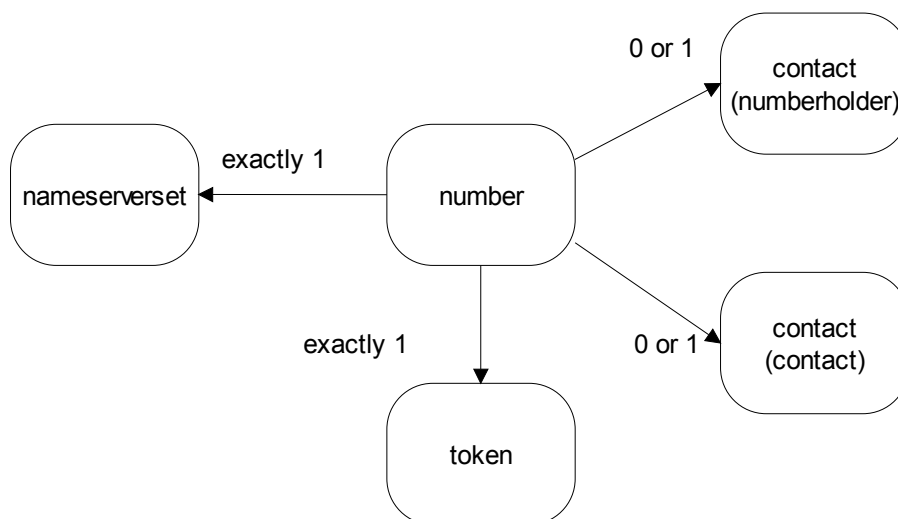
5 Registry object types

The registry supports the following object types:

- number
- contact
- nameserverset
- token

5.1 Object relation

Objects are related to each other as follows:



- A number object may refer to a numberholder (contact type object)
- A number object may refer to a contact (contact type object)
- A number object must refer to (or contain) a token type object
- A number object must refer to a nameserverset type object
- A contact object may be referenced by several number or nameserverset type objects
- A nameserverset object may be referenced by several number type objects
- A token object for a certain number must only be referenced by a number object for the same number.

5.2 Object ownership and access

All objects are initially owned by the creating registrar. Only the current owner may use (refer to), update, and delete them. Object ownership change is only supported for number type objects, the transaction „**transfer number**“ providing number ownership change is the only transaction which may be applied to „foreign“ objects. A „**transfer number**“ can be interpreted as a combination of a „**delete number**“ (acting on a foreign object) immediately followed by a „**create number**“.

5.3 Object types

5.3.1 Number type object

A „number“ type object describes an E.164 number (and a corresponding ENUM domain name).

5.3.1.1 Attributes

A number object instance contains the following attributes:

<i>Attribute name</i>	<i>type</i>	<i>Cardinality</i>	<i>Description / Comment</i>
e164number	E.164plusString	1	Holds a full qualified E.164 number, including a leading „plus“ (+) sign. Is the primary key of the object
Numberholder	Contact roid (type == person or type == organization)	0 or 1	Contains the roid of a person object, describing the number holder of the current number
Contact	Contact roid (type == person or type == role)	0 or 1	Contains the roid of a person object, describing the contact person for the number
Nameserverset	Nameserverset roid	1	Contains the roid of the nameserverset object to which the number is to be delegated.
Numberrangeholder	NRHId	0 or 1	Not required/applicable for enum.nl
Whitepages	Flag	1	Not applicable for enum.nl
directory_assistance	Flag	1	Not applicable for enum.nl
online_directory	Flag	1	Not applicable for enum.nl

5.3.1.2 Attribute policy

Any number object provisioned with the registry must fulfill the following policy requirements:

- the **e164number** attribute must successfully be associated with a kind of number known to the registry.
- If given, the **numberholder** attribute must contain a roid of an existing contact object, which may be either of type „person“ or type „organization“. The roid of a „role“ type object must not be used as numberholder. The referenced object must be owned by the same registrar as the current number object.
- If given, the **contact** attribute must contain a roid of an existing contact object of type „person“ or „role“, owned by the same registrar. A „organization“ type contact is not allowed here.
- The **nameserverset** attribute must contains a roid of an existing nameserverset object, owned by the same registrar as the number object.
- If given, the **whitepages** flag must be either „true“, „false“, „0“ or „1“. Default: „false“. If not given, defaults to „true“.
- If given, the **directory_assistance** flag must be either „true“, „false“, „0“ or „1“. Default: „false“. If not given, defaults to „true“.

- If given, the **online_directory** flag must be either „true“, „false“, „0“ or „1“. Default: „false“. If not given, defaults to „true“.

Please note that the attributes „whitepages“, „directory_assistance“ and „online_directory“ are not used for ENUM delegations under 1.3.e164.arpa. However, those attributes need to be present in requests in order to adhere to the XML schema and should currently be set to „false“.

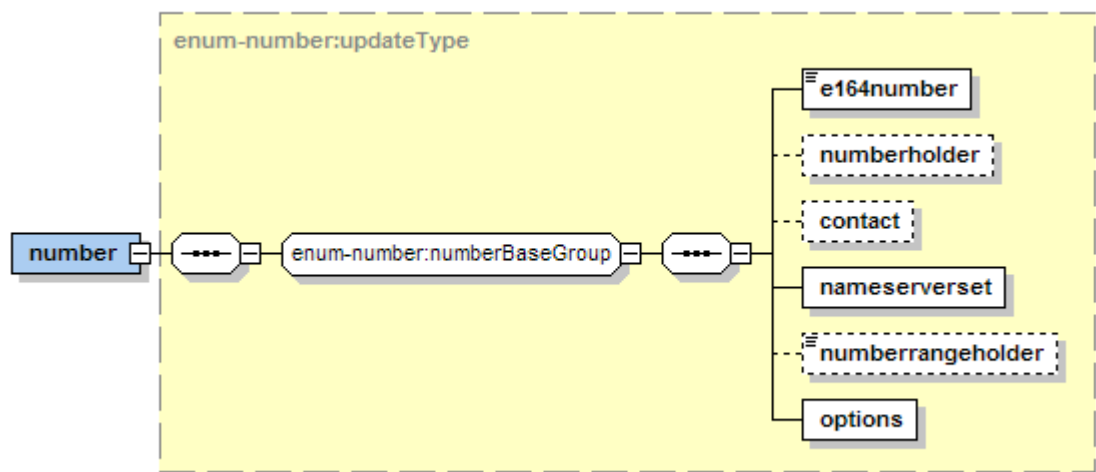
Possible error / warning conditions related to the attribute policy:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000010	Error	Unknown kind of number
EN000011	Error	Contact does not exist
EN000012	Error	Contact owned by different client
EN000013	Error	Numberholder contact is of wrong type
EN000014	Error	Nameserverset does not exist
EN000015	Error	Nameserverset owned by different client
EN000016	Error	Unknown number range holder

Note: Several conditions are expected to be caught by the XML schema check, eg the flag names and their values.

5.3.1.3 Formal syntax (XML schema)

The XML schema of a number object is defined in the Schema file „enum-number-1.0.xsd“, its identifying namespace is „<http://rxsd.enum.nl/enum-number-1.0>“. A graphical representation of the schema follows:



5.3.1.4 Example

The example below does not include namespace declarations, may include invalid data, and is just included for demonstration purposes:

```
<number>
  <e164number>+31201234567</e164number>
  <numberholder>C00012345-ENUMNL</numberholder>
  <contact>C00012346-ENUMNL</contact>
  <nameserverset>N00004711-ENUMNL</nameserverset>
  <numberrangeholder>22</numberrangeholder>
  <options whitepages="true" directory_assistance="true"
online_directory="true">
</number>
```

5.3.2 Contact type object

A „contact“ type object describes a contact, and holds information about a individual person, a legal entity (e.g. Company) or a role.

5.3.2.1 Attributes

An instance of an „contact“ object contains the following attributes („MUST“ indicates that an attribute is mandatory, „MAY“ indicates an optional attribute. „NULL“ indicates that the attribute must not be given, other strings indicate required contents of the attribute):

<i>Attribute</i>	<i>Description</i>	<i>„person“ subtype</i>	<i>„organization“ subtype</i>	<i>„role“ subtype</i>
Roid	Contains the roid of the object. Used to identify an contact object (primary key)	MUST	MUST	MUST
Type	Contains a string, describing the contact subtype	„person“	„organization“	„role“
organization	Contains the full name of an organization	MAY	MUST	MAY
Commercial-register-number	Contains the organization's commercial register identification (number)	MAY	MAY	MAY
Title	Contains a person's title	MAY	NULL	MAY
Firstname	Contains a person's first name(s)	MUST	NULL	MAY
Lastname	Contains a person's last name(s)	MUST	NULL	MUST (is role)
Address	Contains the address of the contact.	MAY	MUST	MAY
		Streetname		MUST
		Streetnumber		MAY
		Apartment		MAY
		Postalcode		MUST
		City		MUST
		State		MAY
Country		MUST		
Phone	Contains the phone number of the contact	MAY	MUST	MAY
Fax	Contains the fax number of the contact	MAY	MAY	MAY
Email	Contains the email address of the contact.	MUST	MUST	MUST

(Remark: Subtable of „address“ row shows requirement within address block, address block may be left out completely)

The types of contact described above describe different types of entities. The „person“ subtype provides information about a human person, which can be expected to be reachable for personal communication

(mail, phone, etc.). The „**organization**“ subtype provides information about a legal entity, which is not a human person. The „**role**“ subtype describes a entity which is responsive to personal communication, but may not consist of a single person (callcenters, noc, etc.)

Use as a „numberholder“

A contact used in the context of „number holder“ describes data associated to the registrant. Only „person“ or „organization“ type contacts may be used as „number holder“. Additional constraints besides the constraints described in the object attribute table may apply, which may in turn vary by kind of number for which a certain object is used as number holder.

Use as a „contact“

A contact used in the context of „contact“ for a certain number describes a contact person or role for a certain number. Only „person“ or „role“ type contacts may be used, since it doesn't make sense to contact an organization as a whole in that case.

5.3.2.2 Attribute policy

Any contact object provisioned with the registry must fulfill the following policy requirements:

- The **roid** attribute must be unique among all objects provisioned in an instance of the registry system. This is ensured by the registry system, since it is not possible to specify a roid on object creation.
- The **type** attribute must not contain any other string than those listed in the table above
- According to the table above, attributes required by the given **type** must be given, and must not be empty
- According to the table above, attributes not allowed by the given **type** must not be given
- If an **address** block is given, the address block attributes must be checked against the requirements as indicated in the table above (depending on the given **type**)
- If an address is given, the **country** attribute must be a valid ISO3166-1 country tag.
- Additional policies may apply depending on the use of the contact.

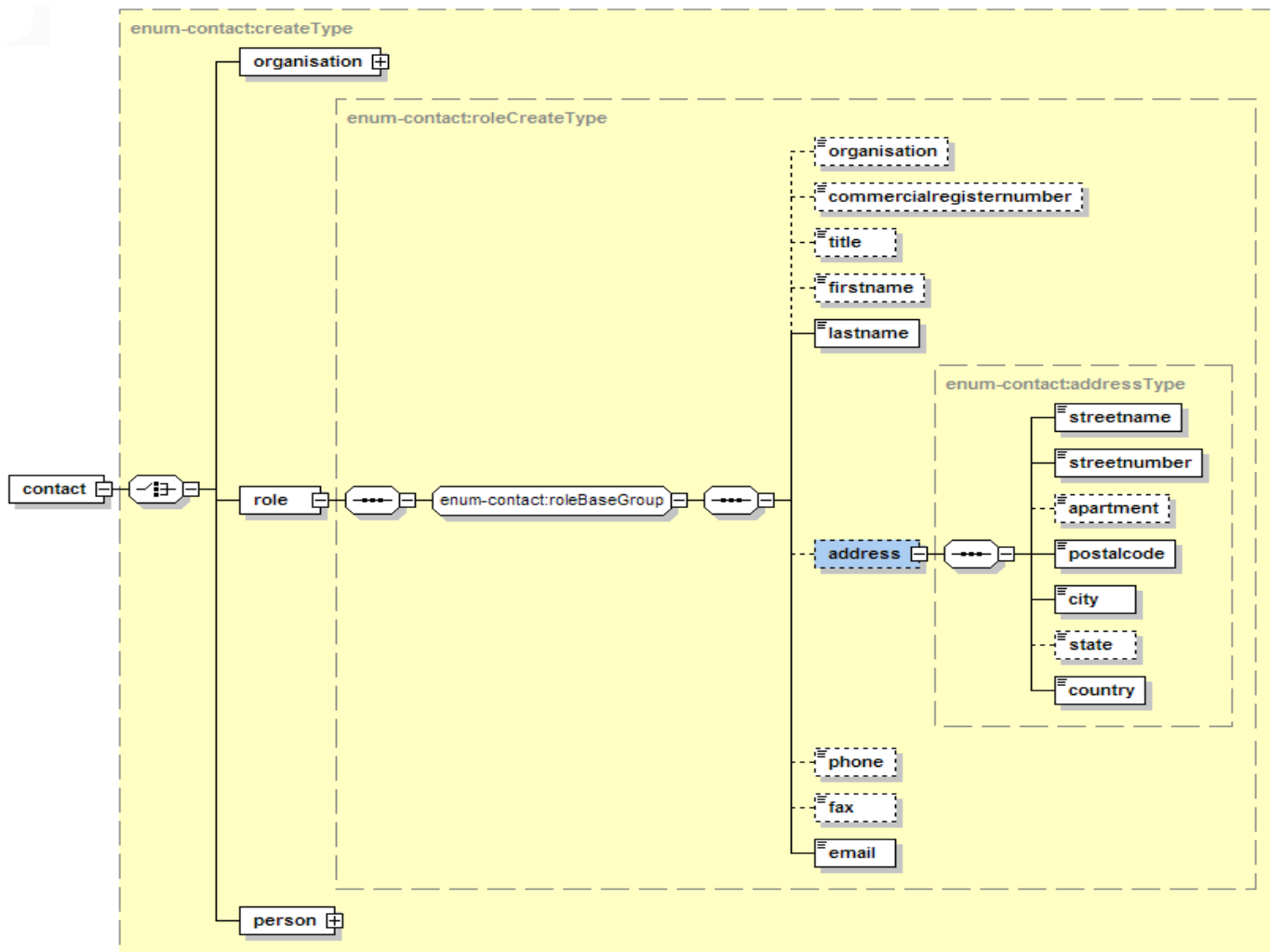
Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000017	Error	Unknown country
EN000060	Error	Invalid character

Note: attribute requirements are expected to be checked by the XML schema validation. They are not listed in the table above.

5.3.2.3 Formal syntax (XML schema)

The XML schema of a contact object is defined in the Schema file „**enum-contact-1.0.xsd**“, its identifying namespace is „<http://rxsd.enum.nl/enum-contact-1.0>“. A graphical representation of the schema for each contact type follows:



5.3.2.4 Example

The example shows a „person“ type contact. Namespaces are not shown, and data included in the example may be invalid as the examples is just shown for demonstration.

```
<person>
  <roid>C00001234-ENUMNL</roid>
  <title>Dhr.</title>
  <firstname>Danny</firstname>
  <lastname>Umy</lastname>

  <address>
    <streetname>Dorpsplein</streetname>
    <streetnumber>1</streetnumber>
    <apartment>a</apartment>
    <postalcode>1010AA</postalcode>
    <city>Het Dorp</city>
    <country>NL</country>
  </address>
  <email disclose="false">D.Umy@enum.nl</email>
</person>
```

5.3.3 Nameserverset type object

A „nameserverset“ type object describes a set of nameservers. A nameserverset is referenced by a number object, and describes the set of nameservers to which the domain corresponding to a certain number is delegated.

5.3.3.1 Attributes

A nameserverset object contains the following attributes:

<i>Attribute name</i>	<i>Type</i>	<i>Cardinality</i>	<i>Description</i>
roid	roid	1	Contains nameserverset roid (auto-generated by server)
hostname	hostname	2 – 5	Contains fqdn (fully qualified domain name) of name servers
contact	contact roid	0 – 1	Contains reference to a contact object

- The **roid** („repository object identifier“) attribute serves as a unique object identifier of a certain nameserverset. It is used to reference nameserverset objects from number object. The roid is auto-generated by the server at the time the nameserverset is generated, and cannot be changed
- The **hostname** attributes contains fqdn's (fully qualified domain names) of name servers. Those name servers will be the target of DNS delegations when the name server set is used in a number object. A minimum of 2 distinct name servers are required, the maximum number of supported name servers is 5.
- The **contact** attribute may be used to reference to a contact object. Information contained in this contact object is considered to describe a contact person who is responsible for operation of the nameservers.

5.3.3.2 Attribute policy

Any nameserverset object provisioned with the registry must fulfill the following policy requirements:

- the **roid** attribute must contains a object identifier which is unique among all objects provisioned in the current instance of the registry.
- The **hostname** attributes must contain strings which contain valid DNS labels (no IDN hostnames) and end in a valid TLD. The hostname must not be „below“ the domain managed by the registry instance because glue is not supported.
- No two **hostname** attributes must contains the same hostname.
- If given, the **contact** attribute must contain the roid of a contact object which is owned by the same registrar as the current nameserverset. The contact must be of type „person“ or „role“.

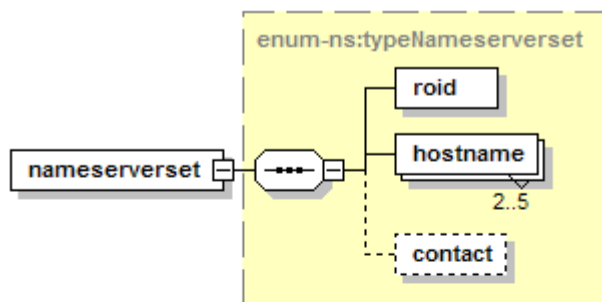
Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000018	Error	Invalid hostname
EN000019	Error	No glue allowed
EN000020	Warning	Duplicate hostname
EN000021	Error	Hostname does not resolve
EN000011	Error	Contact does not exist
EN000012	Error	Contact owned by different client

(Note: XML schemas are expected to catch more errors, but those should yield EN000004 (XML Schema error)

5.3.3.3 Formal syntax (XML schema)

The XML schema of a contact object is defined in the Schema file „**enum-nsset-1.0.xsd**“, its identifying namespace is „<http://rxsd.enum.nl/enum-nsset-1.0>“. A graphical representation of the schema follows:



5.3.3.4 Example

(Excerpt from an „info“ command performed on a nameserverset object, without namespace, containing probably invalid example data):

```
<nameserverset>
  <roid>N00012345-ENUMNL</roid>
  <hostname>ns01.enum.nk</hostname>
  <hostname>ns02.enum.nl</hostname>
  <hostname>ns03.enum.nl</hostname>
  <contact>C00004711-ENUMNL</contact>
</nameserverset>
```

5.3.4 Token type object

A „token“ type object asserts the right to use on a number. Tokens are never used „standalone“ in transactions, they are always used as part of a command modifying a number object. It contains only data necessary to audit that validation, initiate recurring validation etc. **It must not be used to transport additional data beyond that purpose.**

Additional documentation can be found in <http://www.ietf.org/rfc/rfc5105.txt>.

5.3.4.1 Attributes

<token>	This section holds the validation token. The token contains two mandatory sections (validation, signature) and one optional section (tokendata).
<validation serial="xxxxx">	This section contains the validation “meta” data and is mandatory. The „serial“ parameter must be set by the VE to uniquely identify its validation tokens.
<e164number>	full qualified E.164 number for which validation was carried out.
<validator>	numeric (registry) ID of the VE
<registrarid>	numeric (registry) ID of the registrar
<method>	numeric ID of the validation method used
<createdate>	Date when the Token was created/signed
<expiredate>	Date when the Token will expire and a revalidation is necessary
<tokendata>	This section is used to store Data contained in the token, eg. Contact data to be used during revalidation
<Signature>	The signature section contains the xml-signature. Everything inside the <Token> section has to be signed. The signature is a enveloped signature based on the standards of W3C.

5.3.4.2 Attribute policy

- the **validator** must be known to the registry
- the **registrarid** must match the registrarid of the corresponding number object
- the **method** id must be known to the registry, and must be valid for the kind of number affected.
- the **signature** must be valid.

5.3.4.3 Formal syntax (XML schema)

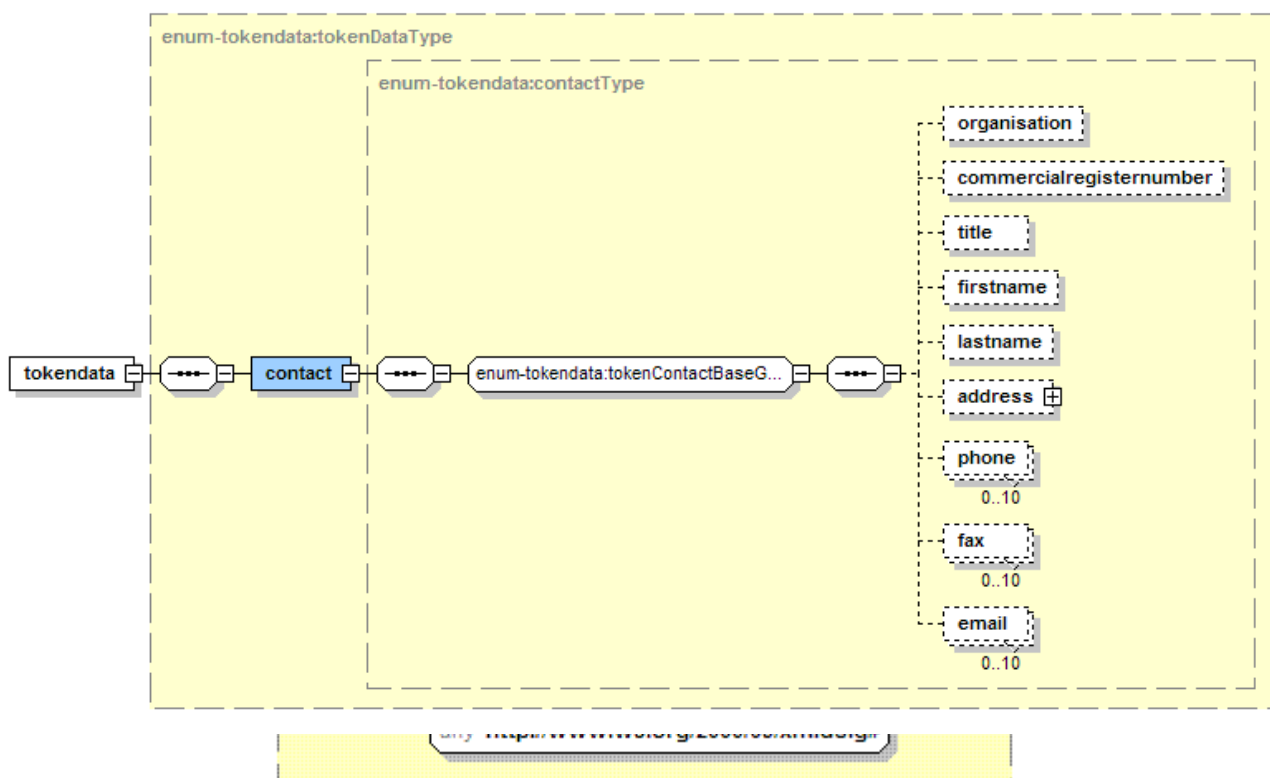
The validation token is defined in two separate schemas:

The validation token core scheme:

Namespace: <http://rxsd.enum.nl/enum-token-1.0>, File „enum-token-1.0.xsd“

The token data scheme:

Namespace: <http://rxsd.enum.nl/enum-tokendata-1.0.xsd>, File: „enum-tokendata-1.0.xsd“.



5.3.4.4 Example

```
<token ID="signed">
  <validation serial="23454">
    <e164number>+31201234567</e164number>
    <validator>4711</validator>
    <registrarid>0815</registrarid>
    <method>22</method>
    <createdate>2004-01-01</createdate>
    <expiredate>2005-01-01</expiredate>
  </validation>
  <tokendata>
    <contact>
      <firstname>Danny</firstname>
      <lastname>Uummy</lastname>
    </contact>
  </tokendata>
  <Signature>
    <!-- Here comes the signature -->
  </Signature>
</token>
```

6 EPP commands and messages

6.1 Transactions (transform commands)

6.1.1 create number

Transaction:	CREATENUMBER	
Description	Delegate the ENUM domain for the specified E.164 number.	
Preconditions:	Token:	<ul style="list-style-type: none"> The token has to pass the generic token verification The create timestamp of the token must be within a window of 10 days in the past to 1 day in the future.
	Nameserverset:	<ul style="list-style-type: none"> See transaction policy below A nameserver-check is not performed.
	Locks:	<ul style="list-style-type: none"> Acquire Write locks on the number and all enveloping prefixes. Acquire Read locks on the contacts involved.
Postconditions	<ul style="list-style-type: none"> The number and the token are entered into the database. The accounting data is modified. Nameserver updates are queued/triggered. 	
Remarks	<ul style="list-style-type: none"> We don't do complex operations. Person objects have to be created before they are referenced in a delegation request. 	

6.1.1.1 Create number transaction policy

In addition to the number object attribute policy and the generic transaction policy, the following policy must be fulfilled:

- The **client** must be allowed to perform transactions for the affected kind of number.
- The **e164number** in question must not yet exist, neither must a more or a less specific number exist
- The **token** found in the command must pass the generic token verification requirements
- The **e164number** found in the **token** must match the e164number attribute in the number object.
- The **registrar id** found in the **token** must match the registrar id of the client invoking the transaction.
- The **numberholder** object referenced must pass the requirements of the affected kind of number, this requirement may be affected by the flags set on the number (whitepages entry etc.)

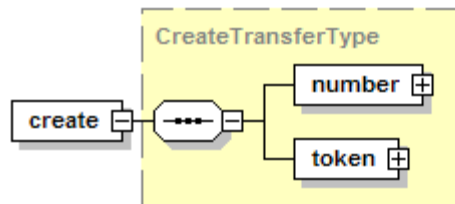
Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000022	Error	Number already exists
EN000031	Error	Permission denied on kind of number
EN000023	Error	Number blocked by less specific
EN000024	Error	Number blocked by more specific
EN000025	Error	Token created for different client
EN000026	Error	Numberholder does not fulfill criteria for affected kind of number
EN000027	Error	Number does not match with token

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000028	Error	Numberholder not given

(Note: additional conditions from the generic transaction policy, the number object policy, the generic token verification policy plus the number range specific policy may apply, but are not shown in table above)

6.1.1.2 create number epp transaction schema



6.1.1.3 create number epp transaction example

Client command:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <create>
      <enum-number:create xmlns:enum-number="http://rxd.enum.nl/enum-number-1.0"
        xsi:schemaLocation="http://rxd.enum.nl/enum-number-1.0
        enum-number-1.0.xsd">
        <enum-number:e164number>+31201234567</enum-number:e164number>
        <enum-number:numberholder>C00012345-ENUMNL
          </enum-number:numberholder>
        <enum-number:contact>C00004711-ENUMNL</enum-number:contact>
        <enum-number:nameserverset>N00012312-ENUMNL
          </enum-number:nameserverset>
        <enum-number:numberrangeholder>22</enum-number:numberrangeholder>
        <enum-number:options whitepages="false" directory_assistance="false"
          online_directory="false" disabled="false" />
        <enum-token:token xmlns:enum-token="http://rxd.enum.nl/enum-token-1.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://rxd.enum.nl/enum-token-1.0
          enum-token-1.0.xsd" ID="Signed">
          <enum-token:validation serial="23454">
            <enum-token:e164number>+31201234567
              </enum-token:e164number>
            <enum-token:validator>22</enum-token:validator>
            <enum-token:registrarid>0815</enum-token:registrarid>
            <enum-token:createdate>2004-01-01</enum-token:createdate>
            <enum-token:expiredate>2005-01-01</enum-token:expiredate>
          </enum-token:validation>
          <enum-tokendata:tokendata
            xmlns:enum-tokendata="http://rxd.enum.nl/enum-tokendata-
            1.0"
            xsi:schemaLocation="http://rxd.enum.nl/enum-tokendata-1.0
            enum-tokendata-1.0.xsd" >
            <enum-tokendata:contact>
              <enum-tokendata:firstname>Danny</enum-tokendata:firstname>
              <enum-tokendata:lastname>Umyy
                </enum-tokendata:lastname>
            </enum-tokendata:contact>
          </enum-tokendata:tokendata>
          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

```

```

                <!-- Here comes the signature -->
            </Signature>
        </enum-token:token>
    </enum-number:create>
</create>
<clTRID>ABC-12345</clTRID>
</command>
</epp>

```

Server response:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
    xmlns:si="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <enum-number:creData xmlns:enum-number
        xmlns:enum-number="http://rxd.enum.nl/enum-number-1.0"
        xsi:schemaLocation="http://rxd.enum.nl/enum-number-1.0
          enum-number-1.0.xsd">
        <enum-number:e164number>+31201234567</enum-number:e164number>
        <enum-number:crDate>2004-10-14T12:00:42.0Z</enum-number:crDate>
        <enum-number:exDate>2005-01-01T23:59:59.9Z</enum-number:exDate>
      </enum-number:creData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>ENUMNL-200410141234</svTRID>
    </trID>
  </response>
</epp>

```

6.1.2 update number

Transaction:	UPDATENUMBER	
Description	Change the delegation data and contact references for specific E.164 number.	
Preconditions:	Attributes:	<ul style="list-style-type: none"> See transaction policy below
	Locks:	<ul style="list-style-type: none"> Write lock on the number. Read lock on all referenced (new) contacts.
Postconditions	<ul style="list-style-type: none"> The changes are made to the database. The accounting data is modified. Nameserver updates are being queued/triggered if necessary. White pages entries are updated (if necessary) 	
Remarks	<ul style="list-style-type: none"> We don't allow tokens in UPDATENUMBER. Tokens must be explicitly updated using RENEWNUMBER 	

6.1.2.1 Update number transaction policy

In addition to the number object attribute policy and the generic transaction policy, the following policy requirements must be fulfilled:

- The **client** must be allowed to provision transactions for the kind of number affected.

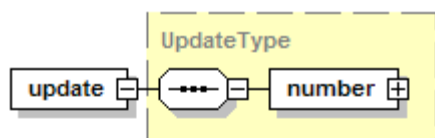
- The **e164number** attribute must describe an already existing object which is owned by the requesting registrar.
- The given **numberholder** must fulfill the numberholder policy of the affected kind of number, possibly influenced by the number's flags.

Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000029	Error	Number does not exist
EN000030	Error	Number belongs to different client
EN000030	Error	Permission denied on kind of number
EN000026	Error	Numberholder does not fulfill criteria of affected kind of number.

(Note: additional conditions from the generic transaction policy, the number object policy, the number range specific policy may apply, but are not shown in table above)

6.1.2.2 update number epp transaction



6.1.2.3 update number epp transaction example

Client command:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <update>
      <enum-number:update xmlns:enum-number="http://rxsd.enum.nl/enum-number-1.0"
        xsi:schemaLocation="http://rxsd.enum.nl/enum-number-1.0
        enum-number-1.0.xsd">
        <enum-number:number
          <enum-number:options whitepages="false" online_directory="false"
          online_directory="false" disabled="false">
            <enum-number:e164number>+509876543</enum-number:e164number>
            <enum-number:numberholder>C00223344-ENUMNL</enum-number:numberholder>
            <enum-number:contact>C00443322-ENUMNL</enum-number:contact>
            <enum-number:nameserverset>N00001122-ENUMNL</enum-
          number:nameserverset>
            <enum-number:numberrangeholder>1234
            </enum-number:numberrangeholder>
            <enum-number:options whitepages="false" online_directory="false"
          online_directory="false" />
        </enum-number:update>
      </update>
      <clTRID>ABC-12345</clTRID>
    </command>
  </epp>
  
```

Server response

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:si="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>ENUMNL-200410141254</svTRID>
    </trID>
  </response>
</epp>
```

6.1.3 delete number

Transaction:	DELETENUMBER	
Description	Deletes the ENUM domain for the specified E.164 number.	
Preconditions:	Policy:	• See below for transaction policy
	Locks:	• Write lock on number.
Postconditions	<ul style="list-style-type: none"> the domain delegation is deleted, nameserver updates (deletion of records) are queued. Certain number kinds may impose a „cool down“ period on delegations. See below for details If necessary, external data updates (white pages) are triggered. 	
Remarks	<ul style="list-style-type: none"> Only instantaneous deletes are supported. As soon as a DELETENUMBER transaction is processed, the number is deleted. 	

Note on „cooldown period“: In certain number ranges, a so called „cool down period“ may be imposed on numbers when deleted. For numbers within those ranges, a successful „delete number“ does not immediately remove the number delegation from the registry. In those cases, only corresponding DNS entries are removed, the number is excluded from invoicing, and is queued for „final deletion“ (which usually takes place a few weeks in the future). In this „cooldown period“, the number cannot be registered by any client. However, it may be registered again after the cooldown period has expired.

Numbers in ranges without a cooldown period can be registered again immediately after a successful „delete number“.

6.1.3.1 Delete number transaction policy

In addition to the generic transaction policy, the following policy has to be fulfilled for the transaction to proceed:

- The **registrar** must be allowed to perform a DELETENUMBER command for the affected kind of number.
- The **e164number** attribute must describe an object which is owned by the requesting registrar.

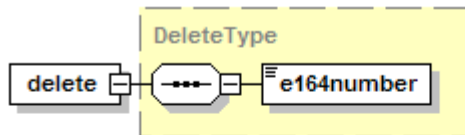
Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000029	Error	Number does not exist
EN000030	Error	Number belongs to different client

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000031	Error	Permission denied on kind of number

(Note: additional conditions from the generic transaction policy may apply, but are not shown in the table above)

6.1.3.2 delete number epp transaction



6.1.3.3 delete number epp transaction example

Client command:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <delete>
      <enum-number:delete xmlns:enum-number="http://rxsd.enum.nl/enum-number-1.0"
        xsi:schemaLocation="http://rxsd.enum.nl/enum-number-1.0
        enum-number-1.0.xsd">
        <enum-number:e164number>+509876543</enum-number:e164number>
      </enum-number:delete>
    </delete>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Server response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:si="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>ENUMNL-200410141266</svTRID>
    </trID>
  </response>
</epp>
```

6.1.4 renew number (supply a new token)

Transaction:	RENEWNUMBER	
Description	The Token for a specified E.164 number is updated, which renews the number's delegation.	
Preconditions:	Policy:	<ul style="list-style-type: none"> See below.
	Locks:	<ul style="list-style-type: none"> Read lock on number.

Postconditions	<ul style="list-style-type: none"> The new Token is added to the system, replacing the previous one The expiry attribute of the number object is adjusted to the expiry date of the new token.
Remarks	<ul style="list-style-type: none"> A new Token replaces the previous one, regardless if the new token expires earlier than the old one.

6.1.4.1 Renew number transaction policy

In addition to the generic transaction policy, the following policies have to be fulfilled for the transaction to proceed:

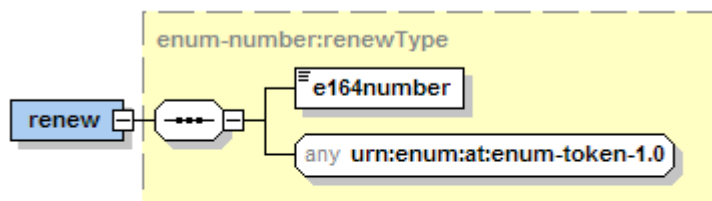
- The **registrar** must be allowed to perform transactions for the kind of number in question
- The **e164number** contained in the number object must refer to an existing number object owned by the requesting registrar.
- The **token** must pass the generic token verification
- The **e164number** contained in the token must match the e164number of the number object given
- The **registrar id** contained in the token must match the requesting client's registrar id.

Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000031	Error	Permission denied on kind of number
EN000029	Error	Number does not exist
EN000030	Error	Number owned by different client
EN000056	Error	Token does not match number

(Note: additional conditions from generic transaction policy, token verification policy and number range policy may apply, but are not shown in above table)

6.1.4.2 renew number epp transaction



6.1.4.3 renew number transaction example

Client command:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <command>
    <renew>
      <enum-number:renew xmlns:enum-number="http://rxd.enum.nl/enum-number-1.0"
        xsi:schemaLocation="http://rxd.enum.nl/enum-number-1.0
          enum-number-1.0.xsd">
        <enum-number:e164number>+31201234567</enum-number:e164number>
        <enum-token:token xmlns:enum-token="http://rxd.enum.nl/enum-
token-1.0"
          xsi:schemaLocation="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://rxd.enum.nl/enum-token-1.0
```

```

        enum-token-1.0.xsd" ID="Signed">
    <enum-token:validation serial="23454">
        <enum-token:e164number>+31201234567
        </enum-token:e164number>
        <enum-token:validator>12</enum-token:validator>
        <enum-token:registrarid>0815</enum-token:registrarid>
        <enum-token:method>222</enum-token:method>
        <enum-token:createdate>2004-01-01</enum-token:createdate>
        <enum-token:expiredate>2005-01-01</enum-token:expiredate>
    </enum-token:validation>
    <enum-tokendata:tokendata
        xmlns:enum-tokendata="http://rxsd.enum.nl/enum-tokendata-
1.0"
        xsi:schemaLocation="http://rxsd.enum.nl/enum-tokendata-1.0
enum-tokendata-1.0.xsd" >
    <enum-tokendata:contact>
        <enum-tokendata:firstname>Danny</enum-tokendata:firstname>
        <enum-tokendata:lastname>Umyy
        </enum-tokendata:lastname>
    </enum-tokendata:contact>
    </enum-tokendata:tokendata>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <!-- Here comes the signature -->
    </Signature>
    </enum-token:token>
    </enum-number:renew>
</renew>
<clTRID>ABC12345</clTRID>
</command>
</epp>

```

Server response:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
    xmlnsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
    <response>
        <result code="1000">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>ABC-12345</clTRID>
            <svTRID>ENUMNL-200410141288</svTRID>
        </trID>
    </response>
</epp>

```

6.1.5 create contact

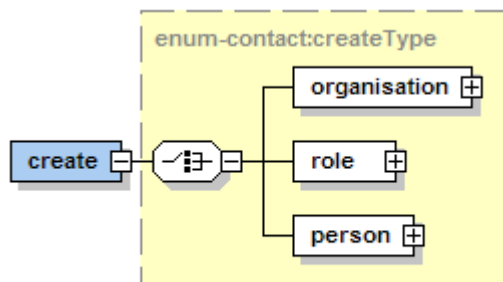
Transaction:	CREATECONTACT	
Description	A contact type object is created	
Preconditions:	Attributes:	<ul style="list-style-type: none"> Contact information delivered in request must fulfill requirements as described in „contact object attribute policy“
	Policy:	<ul style="list-style-type: none"> No restrictions on contact creation.
	Locks:	<ul style="list-style-type: none"> Write lock on object being created.
Postconditions	<ul style="list-style-type: none"> A repository object id is generated The contact is added to the system The client receives the roid of the newly created contact 	

Remarks	<ul style="list-style-type: none"> roid namespace is shared between registrars, so that roids are unique to an instance of the system. roids are auto-generated by the registry
---------	---

6.1.5.1 Create contact transaction policy

Beyond the generic transaction policy and the contact attribute policy, no additional policies have to be fulfilled for the transaction to proceed.

6.1.5.2 create contact epp transaction



6.1.5.3 create contact epp transaction example

Client command:

```

<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <create>
      <enum-contact:create xmlns:enum-contact="http://rxsd.enum.nl/enum-contact-
1.0"
        xsi:schemaLocation="http://rxsd.enum.nl/enum-contact-1.0
enum-contact-1.0.xsd">
        <enum-contact:person>
          <enum-contact:title>Dr.</enum-contact:title>
          <enum-contact:firstname>Joe</enum-contact:firstname>
          <enum-contact:lastname>User</enum-contact:lastname>
          <enum-contact:email>joe.user@example.com</enum-contact:email>
        </enum-contact:person>
      </enum-contact:create>
    </create>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
  
```

Server response:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <enum-contact:creData xmlns:enum-contact
        xmlns:enum-contact="http://rxsd.enum.nl/enum-contact-1.0"
        xsi:schemaLocation="http://rxsd.enum.nl/enum-contact-1.0
enum-contact-1.0.xsd">
        <enum-contact:roid>C00012345-ENUMNL</enum-contact:roid>
      </enum-contact:creData>
    </resData>
  </response>
</epp>
  
```

```

    <enum-contact:crDate>2004-10-14T12:00:42.0Z</enum-contact:crDate>
  </enum-contact:creData>
</resData>
<trID>
  <clTRID>ABC-12345</clTRID>
  <svTRID>ENUMNL-200410141234</svTRID>
</trID>
</response>
</epp>

```

6.1.6 update contact

Transaction:	UPDATECONTACT	
Description	A contact type object is updated	
Preconditions:	Attributes:	<ul style="list-style-type: none"> Contact information delivered in request must fulfill requirements as described in „contact object attribute policy“ Contact update must not break requirements for existing „uses“ of affected contact, see transaction policy below
	Policy:	<ul style="list-style-type: none"> Collect requirements from uses, iterate over requirements. Refuse transaction if any of the requirements cannot be fulfilled. Contact type and contact handle cannot be changed
	Locks:	<ul style="list-style-type: none"> Write lock on contact object, read lock on all objects referencing to that contact
Postconditions	<ul style="list-style-type: none"> The contact is updated Accounting log is written 	
Remarks		

6.1.6.1 Update contact transaction policy

In addition to the generic transaction policy and the contact object attribute policy, the following policies have to be fulfilled for the transaction to proceed:

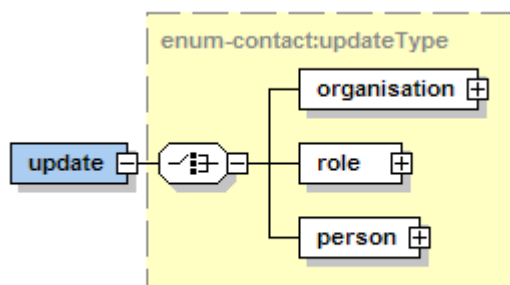
- The **roid** specified in the transaction request must describe an already existing contact which is owned by the requesting registrar.
- For any **number object** referencing the affected contact, the policy required by the kind of number of each affected number object must be fulfilled.
- The **type** of contact must not be changed, even if the update would fulfill requirements of the new contact type.
- The contact object must still fulfill requirements implied by all uses of the contact.

Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000011	Error	Contact does not exist
EN000012	Error	Contact owned by different client
EN000026	Error	Numberholder does not fulfill criteria of affected number kind
EN000032	Error	Cannot change contact type

(Note: additional conditions from generic transaction policy and contact object policy may apply, but are not included in above table)

6.1.6.2 update contact epp transaction schema



6.1.6.3 update contact epp transaction example

Client command:

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <update>
      <enum-contact:update xmlns:enum-contact="http://rxd.enum.nl/enum-contact-
1.0"
        xsi:schemaLocation="http://rxd.enum.nl/enum-contact-1.0
enum-contact-1.0.xsd">
        <enum-contact:person>
          <enum-contact:roid>C00012345-ENUMNL</enum-contact:roid>
          <enum-contact:title>Dr.</enum-contact:title>
          <enum-contact:firstname>Joe</enum-contact:firstname>
          <enum-contact:lastname>User</enum-contact:lastname>
          <enum-contact:email>joe.user@example.com</enum-contact:email>
        </enum-contact:person>
      </enum-contact:update>
    </update>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Server response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:si="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>ENUMNL-200410141234</svTRID>
    </trID>
  </response>
</epp>
```

6.1.7 delete contact

Transaction:	DELETECONTACT	
Description	A contact type object is deleted	
Preconditions:	contact:	• Contact object is not referenced in any other object.
	Policy:	• Roids must never be reused.
	Locks:	• Write lock on contact object.
Postconditions	<ul style="list-style-type: none"> • The contact is deleted • Accounting log is written 	
Remarks		

6.1.7.1 Delete contact transaction policy

In addition to the generic transaction policy, the following policies must be fulfilled for the transaction to proceed:

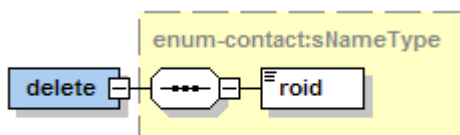
- The **roid** must describe an existing contact which is owned by the requesting registrar
- The **contact** must not be referenced in any number or nameserver set object.

Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000011	Error	Contact does not exist
EN000012	Error	Contact owned by different client
EN000033	Error	Contact still in use

(Note: additional conditions from generic transaction policy may apply, but are not shown in the table above)

6.1.7.2 delete contact epp transaction



6.1.7.3 Delete contact epp transaction example

Client command:

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
epp-1.0.xsd">
  <command>
    <delete>
      <enum-contact:delete xmlns:enum-contact="http://rxsd.enum.nl/enum-contact-
1.0"
        xsi:schemaLocation="http://rxsd.enum.nl/enum-contact-1.0
enum-contact-1.0.xsd">
        <enum-contact:roid>C00001234-ENUMNL</enum-contact:roid>
      </enum-contact:delete>
    </delete>
    <c1TRID>ABC-12345</c1TRID>
  </command>
</epp>
```

Server response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>ENUMNL-200410141234</svTRID>
    </trID>
  </response>
</epp>
```

6.1.8 Create nameserverset

Transaction:	CREATENAMESERVERSET	
Description	A nameserverset type object is created	
Preconditions:	nameserverset:	<ul style="list-style-type: none"> Request must fulfill nameserverset object attribute policies
	Policy:	<ul style="list-style-type: none"> No limits on creation of nameserverset
	Locks:	<ul style="list-style-type: none"> Write lock on prospective nameserverset object, read lock on contact object, if referenced.
Postconditions	<ul style="list-style-type: none"> The nameserverset is created Accounting log is written 	
Remarks	<ul style="list-style-type: none"> Handles are auto-generated 	

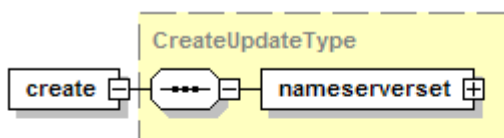
6.1.8.1 Create nameserverset transaction policy

In addition to the generic transaction policy and the nameserverset object attribute policy, the following policies must be fulfilled for the transaction to proceed:

- the **registrar** must be allowed to perform CREATENAMESERVERSET transactions

No additional conditions defined.

6.1.8.2 create nameserverset epp transaction



6.1.8.3 create nameserverset epp transaction example

Client command:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <create>
```

```

<enum-nsset:create
  xmlns:enum-nsset="http://rxsd.enum.nl/enum-nsset-1.0"
  xsi:schemaLocation="http://rxsd.enum.nl/enum-nsset-1.0
enum-nsset-1.0.xsd">
  <enum-nsset:hostname>ns1.example.com</enum-nsset:hostname>
  <enum-nsset:hostname>ns2.example.com</enum-nsset:hostname>
  <enum-nsset:hostname>ns3.example.com</enum-nsset:hostname>
  <enum-nsset:contact>C0001234-ENUMNL</enum-nsset:contact>
</enum-nsset:create>
</create>
<clTRID>ABC-12345</clTRID>
</command>
</epp>

```

Server response:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlnsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <enum-nsset:creData
        xmlns:enum-nsset="http://rxsd.enum.nl/enum-nsset-1.0"
        xsi:schemaLocation="http://rxsd.enum.nl/enum-nsset-1.0
enum-nsset-1.0.xsd">
        <enum-contact:roid>N00012345-ENUMNL</enum-contact:roid>
        <enum-contact:crDate>2004-10-14T12:00:42.0Z</enum-contact:crDate>
      </enum-contact:creData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>ENUMNL-200410141334</svTRID>
    </trID>
  </response>
</epp>

```

6.1.9 Update nameserverset

Transaction:	UPDATENAMESERVERSET	
Description	A nameserverset type object is updated	
Preconditions:	nameserverset:	<ul style="list-style-type: none"> Request must fulfill nameserverset attribute policy requirements If contact is given, it must already exist, and be of contact type „person“ or „contact“
	Policy:	<ul style="list-style-type: none"> Handles of nameserverset must not be updated.
	Locks:	<ul style="list-style-type: none"> Write lock on nameserverset object, read lock on contact object, if referenced. Read locks on all domains which reference the nameserverset
Postconditions	<ul style="list-style-type: none"> The nameserverset is updated The nameserver updated are queued/triggered Accounting log is written 	
Remarks		

6.1.9.1 Update nameserverset transaction policy

In addition to the generic transaction policy and the nameserverset object attribute policy, the following additional policies must be fulfilled for the transaction to proceed:

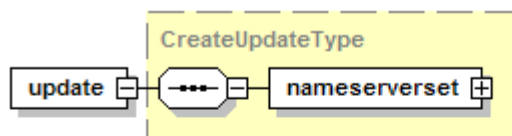
- the **registrar** must be allowed to perform UPDATENAMESERVERSET transactions.
- The **roid** must specify a nameserverset which is owned by the requesting registrar.

Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000014	Error	Nameserverset does not exist
EN000015	Error	Nameserverset is owned by a different client

(Note: additional conditions from the generic transaction policy and the nameserverset object policy may apply, but are not included in the table above)

6.1.9.2 Update nameserverset epp transaction



6.1.9.3 Update nameserverset epp transaction example

Client command:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <update>
      <enum-nsset:update xmlns:enum-nsset="http://rxd.enum.nl/enum-nsset-1.0"
        xsi:schemaLocation="http://rxd.enum.nl/enum-nsset-1.0
        enum-nsset-1.0.xsd">
        <enum-nsset:roid>N1112223-ENUMNL</enum-nsset:roid>
        <enum-nsset:hostname>ns1.example.com</enum-nsset:hostname>
        <enum-nsset:hostname>ns2.example.com</enum-nsset:hostname>
        <enum-nsset:contact>C12340000-ENUMNL</enum-nsset:contact>
      </enum-nsset:update>
    </update>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Server response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:si="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>ENUMNL-200410141234</svTRID>
    </trID>
```

```
</response>
</epp>
```

6.1.10 Delete nameserver set

Transaction:	DELETENAMESERVERSET	
Description	A nameserver set type object is deleted	
Preconditions:	nameserver set:	<ul style="list-style-type: none"> No requirements on the nameserver set itself
	Policy:	<ul style="list-style-type: none"> Nameserver set must not be referenced in any number object.
	Locks:	<ul style="list-style-type: none"> Write lock on nameserver set object
Postconditions	<ul style="list-style-type: none"> The nameserver set is deleted Accounting log is written 	
Remarks		

6.1.10.1 Delete nameserver set transaction policy

In addition to the generic transaction policy, the following additional policies need to be fulfilled for the transaction to proceed:

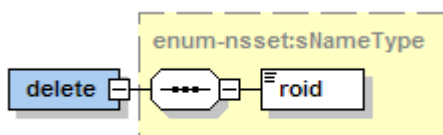
- The **roid** must specify a nameserver set which is not referenced by any number object, and which is owned by the requesting registrar.

Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000014	Error	Nameserver set does not exist
EN000015	Error	Nameserver set owned by different client
EN000034	Error	Nameserver set is still in use

(Note: conditions from the generic transaction policy may apply as well, but are not included in the table above)

6.1.10.2 Delete nameserver set epp transaction



6.1.10.3 Delete nameserver set epp transaction examples

Client command:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <delete>
      <enum-nsset:delete
        xmlns:enum-nsset="http://rxd.enum.nl/enum-nsset-1.0"
```

```

    xsi:schemaLocation="http://rxsd.enum.nl/enum-nsset-1.0
enum-nsset-1.0.xsd">
    <enum-nsset:roid>N00012345-ENUMNL</enum-nsset:roid>
  </enum-nsset:delete>
</delete>
<clTRID>ABC-12345</clTRID>
</command>
</epp>

```

Server response:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:si="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>ENUMNL-200410141234</svTRID>
    </trID>
  </response>
</epp>

```

6.1.11 transfer number

Transaction:	TRANSFERNUMBER	
Description	Transfer the number object for the specified E.164 number to the requesting registrar.	
Preconditions:	Number:	<ul style="list-style-type: none"> Number must be a valid E.164 number Number must be already delegated to a different registrar. Number object must fulfill the number object attribute policies
	Token	<ul style="list-style-type: none"> The token must pass the generic token verification process
	Locks:	<ul style="list-style-type: none"> Write lock on the number. Read lock on all referenced (new) contacts.
Postconditions	<ul style="list-style-type: none"> The number is associated to the new registrar, and updated to reflect the new contact and nameserverset relations. 	
Remarks	<ul style="list-style-type: none"> We don't do complex operations. Person objects and nameserversets have to be created before they are referenced in a delegation request. De facto, this is a DELETENUMBER/CREATENUMBER sequence with another name. The losing registrar will be informed about a successful transfer by a message in his message queue. 	

6.1.11.1 Transfer number transaction policy

In addition to the generic transaction policy and the number object attribute transaction policy, the following policies need to be fulfilled for the transaction to proceed:

- The **client** (registrar) must be allowed to provision transactions for the kind of number in question.
- The **e164number** in question must already exist, and must be owned by a different registrar.

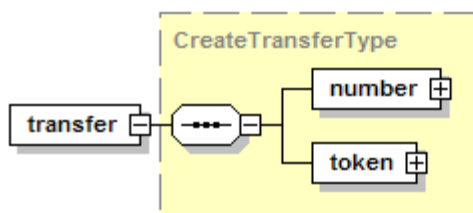
- The **token** found in the command must pass the generic token verification requirements, and must be „fresh“ (validation timestamp between 1 day in the future and 10 days in the past)
- The **e164number** found in the **token** must match the e164number attribute in the number object.
- The **registrar id** found in the **token** must match the registrar id of the client invoking the transaction.
- The **numberholder** object referenced must pass the requirements of the affected kind of number

Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000031	Error	Permission denied on kind of number
EN000029	Error	Number does not exist
EN000035	Error	Number already owned by requesting client
EN000011	Error	Contact does not exist
EN000012	Error	Contact owned by different client
EN000013	Error	Numberholder contact is of wrong type
EN000014	Error	Nameserverset does not exist
EN000015	Error	Nameserverset owned by different client
EN000016	Error	Unknown number range holder
EN000025	Error	Token created for different client
EN000026	Error	Numberholder does not fulfill criteria for affected kind of number
EN000027	Error	Number does not match with token

(Note: additional conditions from the generic token verification policy, the number object policy, a number range specific policy, generic token verification policy may apply, but are not listed in the table above)

6.1.11.2 Transfer number epp transaction



6.1.11.3 Transfer number epp transaction example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <transfer op="request">
      <enum-number:transfer xmlns:enum-number="http://rxsd.enum.nl/enum-number-
1.0"
        xsi:schemaLocation="http://rxsd.enum.nl/enum-number-1.0
enum-number-1.0.xsd">
        <enum-number:e164number>+31201234567</enum-number:e164number>
        <enum-number:numberholder>C0012233-ENUMNL</enum-number:numberholder>
        <enum-number:contact>C00133324-ENUMNL</enum-number:contact>
```

```

<enum-number:nameserverset>N00444334-ENUMNL
  </enum-number:nameserverset>
<enum-number:numberrangeholder>2233
  </enum-number:numberrangeholder>
<enum-number:options whitepages="false" online_directory="false"
online_directory="false" disabled="false" />
  <enum-token:token xmlns:enum-token="http://rxsd.enum.nl/enum-token-1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://rxsd.enum.nl/enum-token-1.0
enum-token-1.0.xsd" ID="Signed">
    <enum-token:validation serial="23454">
      <enum-token:e164number>+31201234567
      </enum-token:e164number>
      <enum-token:validator>234</enum-token:validator>
      <enum-token:registrarid>0815</enum-token:registrarid>
      <enum-token:method>22</enum-token:method>
      <enum-token:createdate>2004-01-01</enum-token:createdate>
      <enum-token:expiredate>2005-01-01</enum-token:expiredate>
    </enum-token:validation>
    <enum-tokendata:tokendata
      xmlns:enum-tokendata="http://rxsd.enum.nl/enum-tokendata-
1.0"
      xsi:schemaLocation="http://rxsd.enum.nl/enum-tokendata-1.0
enum-tokendata-1.0.xsd" >
      <enum-tokendata:contact>
        <enum-tokendata:firstname>Danny</enum-tokendata:firstname>
        <enum-tokendata:lastname>Uummy
        </enum-tokendata:lastname>
      </enum-tokendata:contact>
    </enum-tokendata:tokendata>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <!-- Here comes the signature -->
    </Signature>
  </enum-token:token>
</enum-number:transfer>
</transfer>
<clTRID>ABC-12345</clTRID>
</command>
</epp>

```

Server response:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:si="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <enum-number:creData xmlns:enum-number
        xmlns:enum-number="http://rxsd.enum.nl/enum-number-1.0"
        xsi:schemaLocation="http://rxsd.enum.nl/enum-number-1.0
enum-number-1.0.xsd">
        <enum-number:e164number>+31201234567</enum-number:e164number>
        <enum-number:crDate>2004-10-14T12:00:42.0Z</enum-number:crDate>
        <enum-number:exDate>2005-01-01T23:59:59.9Z</enum-number:exDate>
      </enum-number:creData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>ENUMNL-200410151234</svTRID>
    </trID>
  </response>
</epp>

```

6.2 Query commands

In addition to commands modifying („transforming“) objects, EPP defines „query“ type commands which are used to query the registry for information. No updates to objects are possible using those commands. From the set of commands specified in EPP, the following are supported:

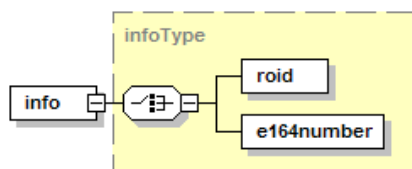
6.2.1 „info“ command

A registrar may issue an „info“ command to receive the currently stored data about an object in the registry. A successful info command response always includes information about exactly one object. An info command can be applied to the following objects/attributes:

- number:e164number (may include token information in response)
- contact:roid
- nsset:roid

Displayed information may be limited depending on client, object etc.

6.2.1.1 „info“ command formal syntax



6.2.1.2 EPP „info“ transaction example

This example queries for a nameserverset object

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <command>
    <info>
      <enum-nsset:info
        xmlns:enum-nsset="http://rxd.enum.nl/enum-nsset-1.0"
        xsi:schemaLocation="http://rxd.enum.nl/enum-nsset-1.0
        enum-nsset-1.0.xsd">
        <enum-nsset:roid>N00012345-ENUMNL</enum-nsset:roid>
      </enum-nsset:info>
    </info>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
  
```

Response:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <enum-nsset:infData
        xmlns:enum-nsset="http://rxd.enum.nl/enum-nsset-1.0"
        xsi:schemaLocation="http://rxd.enum.nl/enum-nsset-1.0
  
```

```

enum-nsset-1.0.xsd">
  <enum-nsset:roid>N00012345-ENUMNL</enum-nsset:roid>
  <enum-nsset:status s="linked"/>
  <enum-nsset:status s="clientDeleteProhibited"/>
  <enum-nsset:hostname>ns1.example.com</enum-nsset:hostname>
  <enum-nsset:hostname>ns2.example.com</enum-nsset:hostname>
  <enum-nsset:contact>C00121212-ENUMNL</enum-nsset:contact>
  <enum-nsset:clID>22</enum-nsset:clID>
  <enum-nsset:crID>22</enum-nsset:crID>
  <enum-nsset:crDate>2004-04-03T22:00:00.0Z</enum-nsset:crDate>
  <enum-nsset:upID>ClientX</enum-nsset:upID>
  <enum-nsset:upDate>2004-10-03T09:00:00.0Z</enum-nsset:upDate>
</enum-nsset:infData>
</resData>
<trID>
  <clTRID>ABC-12345</clTRID>
  <svTRID>54322-XYZ</svTRID>
</trID>
</response>
</epp>

```

6.2.2 „check“ command

The „check“ command can only be applied to „number“ type objects. „check number“ can be used to check for existence of a number object in the registry. In addition to the „exact match“ check, this command as well checks for shorter or longer numbers which would block the number in question. The operation of the „check number“ command is similar to the „finger interface“.

6.2.2.1 „check number“ command example

The example checks for the number „+31 26 3525555“ (ENUM NL contact number).

Request frame:

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?><epp
xmlns="urn:ietf:params:xml:ns:epp-1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <command>
    <check>
      <enum-number:check xmlns:enum-
number="http://rxsd.enum.nl/enum-number-1.0"
xsi:schemaLocation="http://rxsd.enum.nl/enum-number-1.0 enum-number-1.0.xsd">
        <enum-number:e164number>+31263525555</enum-number:e164number>
      </enum-number:check>
    </check>
    <clTRID>111227238528633</clTRID>
  </command>
</epp>

```

Response frame:

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <msgQ count="12" id="553928"/>
    <resData>
      <chkData xmlns="http://rxsd.enum.nl/enum-number-1.0"
xsi:schemaLocation="http://rxsd.enum.nl/enum-number-1.0 enum-number-1.0.xsd">
        <cd>

```

```

        <e164number avail="0">+31263525555</e164number>
        <reason>Less specific, more specific or exact number found</reason>
    </cd>
</chkData>
</resData>
</trID>
    <clTRID>111227238528633</clTRID>
    <svTRID>20050331123305498781-002-enumnl</svTRID>
</trID>
</response>
</epp>

```

6.2.3 „hello“ command

The server responds to the „hello“ command with a greeting similar to the login response. The hello command is the preferred command for EPP link tests, especially if used in short intervals, since it is very lightweight.

6.2.4 „poll“ command

The „poll“ command is used to query and fetch messages from the EPP message queue. A client should regularly poll the EPP server for new messages – recommended interval is 1 hour. A client should poll at least daily to receive notifications in time.

6.2.4.1 „poll“ example

Request (fetching a message):

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?><epp
xmlns="urn:ietf:params:xml:ns:epp-1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
    <command>
        <poll msgID="" op="req"/>
        <clTRID>111227367825442</clTRID>
    </command>
</epp>

```

Response (the fetched message):

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
    <response>
        <result code="1301">
            <msg>Command completed successfully; ack to dequeue</msg>
        </result>
        <msgQ count="10" id="553930">
            <qDate>2005-03-31T09:49:17.19Z</qDate>
            <msg>MT000003: 2 numbers expired</msg>
        </msgQ>
        <resData>
            <message xmlns="http://rxsd.enum.nl/enum-message-1.0" type="number-expired"
xsi:schemaLocation="http://rxsd.enum.nl/enum-message-1.0 enum-message-1.0.xsd">
                <numberList xmlns="http://rxsd.enum.nl/enum-number-1.0"
xsi:schemaLocation="http://rxsd.enum.nl/enum-number-1.0 enum-number-1.0.xsd">
                    <number>
                        <e164number>+315555</e164number>
                        <exDate>2005-03-03T00:00:00.00Z</exDate>
                    </number>
                    <number>
                        <e164number>+31999</e164number>
                        <exDate>2005-01-30T00:00:00.00Z</exDate>
                    </number>
                </numberList>
            </message>
        </resData>
    </response>
</epp>

```

```

    </resData>
    <trID>
      <clTRID>111227367825442</clTRID>
      <svTRID>20050331125437157204-002-enumnl</svTRID>
    </trID>
  </response>
</epp>

```

Request (acknowledging and deleting a message):

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?><epp
xmlns="urn:ietf:params:xml:ns:epp-1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <command>
    <poll msgID="553930" op="ack"/>
    <clTRID>111227367825442</clTRID>
  </command>
</epp>

```

Response (to ack/delete):

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <msgQ count="9" id="553931"/>
    <trID>
      <clTRID>111227367825442</clTRID>
      <svTRID>20050331125437265807-002-enumnl</svTRID>
    </trID>
  </response>
</epp>

```

6.3 EPP response extension

The registry system provides an extension to the standard EPP response format ([RFC3730](#), 2.6). The extension provides a mechanism to include full information about conditions which occurred while processing the command in question. In any case, the registry returns responses including exactly one standard EPP error code, but the response may additionally include several condition codes in the extension.

6.3.1 condition description

A condition contains the following data fields:

- **code**: an alphanumeric identifier of the condition type. Similar conditions, but eg. from transactions on different objects return the same condition code.
- **severity**: The severity of the condition, a text field containing one of the following strings:
 - **info** – the condition is strictly informational, no client action is required. Conditions indicating success could use that severity.
 - **warning** – the condition indicates a problem with the request, but the problem did not make the command fail. The client should investigate the problem, and try to avoid the issue in future requests. Transactions where eg. whitespace has been stripped from certain object attributes could yield a condition of this type.
 - **error** – the condition indicates a problem with the request leading to command failure. The client should assume a problem with his request, and must investigate and solve the problems indicated by the condition – the command did not affect any object in the registry.

- **fatal** – the condition indicates a problem with the registry server. The client should assume no problems with his request, and should try to repeat the request after a few minutes. If the problem persists, the client should contact registry staff.
- **msg**: A human readable error message with a predefined content per **code**. May include variables which are substituted with eg. object identifiers. Clients should assume that **msg** content associated to a **code** does not change (with the exception of variable replacement).
- **details**: human readable details about the condition containing free-formed text. A client should assume that **details** content does change even for conditions with equal **code** depending on the actual situation.

6.3.2 response example

The following example indicates a failed „update contact“ command, and contains two conditions using the extension described above (note: condition codes are just examples):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <response>
    <result code="2303">
      <msg lang="en">Object does not exist</msg>
    </result>
    <extension>
      <conditions
        xmlns="urn:enum:nl:enum-result-1.0"
        xsi:schemaLocation="urn:enum:nl:enum-result-1.0
        enum-result-1.0.xsd">
        <condition code='E0123' severity='fatal'>
          <msg>Person object 'ENUMNL-RT2341' does not exist</msg>
          <details>The person object to be updated does not exist - unable to
            update non-existent objects.</details>
        </condition>
        <condition code='E0124' severity='warning'>
          <msg>Unknown country</msg>
          <details>The country 'Netherlands' as specified in the 'country'
            attribute is unknown to the registry, but is considered to be a
            frequent typo - it has been replaced by 'Netherlands'</details>
        </condition>
      </conditions>
    </extension>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

6.4 EPP message queue

The registry supports EPP message queue commands as specified in [RFC3730](#), 2.9.2.3. The standard “poll” command as described in the RFC is used to fetch and dequeue messages. A registrar should poll the message queue regularly, at least daily.

6.4.1 Message format

Messages are formatted according to [RFC3730](#). The “msg”-Tag of every “msgQ” section contains a human-readable summary of the message contents. Specific message content is delivered in the “resData” section of the message – this section envelopes a custom “message” section, which in turn contains message-type-specific data.

6.4.1.1 Message example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
  epp-1.0.xsd">
  <response>
    <result code="1301">
      <msg>Command completed successfully; ack to dequeue</msg>
    </result>
    <msgQ count="5" id="12345">
      <qDate>2000-06-08T22:00:00.0Z</qDate>
      <msg>MT000000: generic enum.nl message example</msg>
    </msgQ>
    <resData>
      <message
        xmlns="http://rxd.enum.nl/enum-message-1.0"
        xsi:schemaLocation="http://rxd.enum.nl/enum-message-1.0
        enum-message-1.0.xsd" type="example-message">

        [ ... message specific data excluded ... ]

      </message>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

6.4.2 Queue policy

Messages are queued on the server for 21 days, calculated from the day the message was queued. For messages not dequeued by the client within those 21 days, delivery via email is attempted, messages are subsequently dequeued. If a message is delivered by email, messages are reformatted as follows:

- The **mail subject** is built from a fixed prefix specifying the registry instance (e.g. “[enum.nl registry]”, followed by the message queue id prefixed by a “#”-sign, followed by the contents of the “msgQ” / “msg” tag. An example subject could look like:
[enum.nl registry] #4711 MT000001: number +31123 transferred away
- The **mail body** is built from the “message” section of the original message, prefixed by a XML document specification to make a standalone XML document.
- The **from address** and the **SMTP envelope from** is set to a registry administrative staff address, e.g. “support@enum.nl”
- The **to address** to send the mail is set to the technical contact information of the registrar.
- Each message from the queue is sent as a separate mail

6.4.3 Message type definitions

6.4.3.1 epp-late-response (MT0001)

This message is queued for the client either when the response to a previous transaction was not delivered properly to a client (eg. because the connection broke down) or when the final result of a pending transaction is available. In both cases, the response is provided in the message itself.

Only the following commands are considered for this message: “create”, “update”, “delete”, “transfer”, “renew”. No messages are queued for “info”, “hello”, “login”, “logout” commands.

- **msg:** MT000001: EPP response to command with clTRID [<clTRID>] and svTRID [<svTRID>]

- **message content:** an EPP response frame
- **message content example:**

```
<message
  xmlns="http://rxsd.enum.nl/enum-message-1.0"
  xsi:schemaLocation="http://rxsd.enum.nl/enum-message-1.0
enum-message-1.0.xsd" type="epp-late-response">
  <epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
epp-1.0.xsd">
    [ ... epp response frame excluded ... ]
  </epp>
</message>
```

6.4.3.2 number-transferred (MT000002)

This message is queued for the client representing the losing registrar when a number which is under his control is transferred to a different client. The message is queued immediately on execution of the transfer. Each single transfer initiates one message of this type.

- **msg:** MT000002: number <e164number> transferred away
- **message content:** a “trnData” section, containing one “e164number” and one “trDate” tag.
- **message content example:**

```
<message>
  <trnData>
    <e164number>+3112345678</e164number>
    <trDate>2004-12-22T09:00:00.0Z</trDate>
  </trnData>
</message>
```

6.4.3.3 number-expired (MT000003)

This message is queued when numbers expire (because the validation token associated with the number object has expired). The message is queued for the client who had the number under his control, and is queued at the time when the registry performs its daily number expiration job. A single message containing a list of expired numbers is queued per registrar. If no numbers expired for a specific registrar, no message is queued – there are no empty lists.

- **msg:** MT000003: <count> numbers expired at <date>
- **message content:** a “numberList” section, containing a list of “number” tags, which in turn contain “e164number” and “exDate” tags
- **message content example:** (containing 2 numbers)

```
<message>
  <numberList>
    <number>
      <e164number>+311234567</e164number>
      <exDate>2004-12-22T09:00:00.0Z</exDate>
    </number>
    <number>
      <e164number>+3198765543</e164number>
      <exDate>2004-12-22T12:32:33.4Z</exDate>
    </number>
  </numberList>
</message>
```

6.4.3.4 number-exp-warn-30 (MT000004)

This message is queued for a specific client if this client has control over numbers which are set to expire in exactly 30 days from the current date. The message is queued at the time when the registry performs its daily expiration job. A single message contains a list of numbers which are about to expire is queued per registrar. If no numbers are about to expire, no message is queued.

- **msg:** MT000004: <count> numbers expiring in 30 days
- **message content:** similar to “number-expired”
- **message content example:** similar to “number-expired”

6.4.3.5 number-exp-warn-7 (MT000005)

This message is similar to the “number-exp-warn-30” with the exception that only numbers which are set to expire in exactly seven days are considered.

- **msg:** MT000005: <count> numbers expiring in 7 days
- **message content:** similar to “number-expired”
- **message content example:** similar to “number-expired”

6.4.3.6 number-exp-warn-1 (MT000006)

This message is similar to the “number-exp-warn-30” with the exception that only numbers which are set to expire by tomorrow are considered.

- **msg:** MT000006: <count> numbers expiring tomorrow
- **message content:** similar to “number-expired”
- **message content example:** similar to “number-expired”

7 Example Transactions

Here are some examples on typical business processes and how they will translate into transactions with the registry.

7.1 EPP login

All transactions (except the simple existence check via finger) use an EPP session. The session needs to be created first by connecting to the registry server, and logging in. EPP is described in chapter 4 'Extensible Provisioning Protocol EPP'.

7.2 Initial object creation

All registrars will need some objects in the registry database which should be generated before any domains are delegated. A description of the "create" commands can be found in chapter 6.1 'Transactions'.

1. CREATE technical contact
2. CREATE nameserver set

7.3 ENUM Domain for a geographic number

This example illustrates registration of a simple geographic number. For more elaborate examples of registration-processes, see chapter 8 "Test Cases".

1. Check if already delegated
 - There are two ways of doing this: Either use the finger interface or the EPP info command.

2. Obtain Token from VE

How this is done depends a lot on the preferences of the registrar. The registrar can make use of an external VE, or assume that role himself.

3. Create nameserver set

If the registrar hosts the zones on his own name servers, the object will already exist and this step can be skipped.

4. Create contact(s)

Create the necessary contact objects (numberholder and contact). If these are existing contacts, this step can be skipped.

5. Create Number / Transfer Number

Compose a valid createnumber EPP command including the Token and using the appropriate handles. Register a new delegation with this information.

8 Test Cases

This is a description of a number of test cases designed to test the basic functionality of the ENUM application. The test cases are explicitly not meant to test all features of the system: the purpose is only to demonstrate the correct (expected) results in a number of scenarios. In a sense, only the “happy path” of each functional process is tested.

The registrar manual is used as basis for these test cases. The following scenarios are tested:

Contacts

Create a new contact

Update attributes of an existing contact

Delete a contact from the database

Nameserversets

Create a new nameserverset

Update attributes of an existing nameserverset

Delete a nameserverset from the database

Numbers

Create number: new ENUM number delegation

Update number: change delegation attributes

Renew number: supply a new token for a delegation

Transfer number: move a delegation from one registrar to another registrar

Delete number: remove the delegation from the database

General

Info: ask for information on an object

Check whether a number is already in the database

Poll for new messages for the registrar

8.1 Contacts

8.1.1 Create a new contact

Purpose: demonstrate how a new contact is entered in the system and a handle is returned.

Steps:

- 8.1.1.1 Define a nonexistent contact of type person
- 8.1.1.2 Compose a CreateContact EPP command
- 8.1.1.3 Send the CreateContact command to the ENUM system
- 8.1.1.4 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The new contact is stored in the database
 - The unique handle of the new contact is returned

8.1.2 Update attributes of an existing contact

Purpose: demonstrate how an existing contact is updated in the system.

Steps:

- 8.1.2.1 Select the handle of an existent contact of type person from the database
- 8.1.2.2 Make some changes to the contact data (name, title and email address)
- 8.1.2.3 Compose an UpdateContact EPP command
- 8.1.2.4 Send the UpdateContact command to the ENUM system
- 8.1.2.5 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The contact is correctly updated in the database

8.1.3 Delete a contact from the database

Purpose: demonstrate how a contact is deleted from the system.

Steps:

- 8.1.3.1 Select the handle of an existent contact of type person from the database
- 8.1.3.2 Compose a DeleteContact EPP command
- 8.1.3.3 Send the DeleteContact command to the ENUM system
- 8.1.3.4 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The contact is correctly deleted from the database

8.2 Nameserversets

8.2.1 Create a new nameserverset

Purpose: demonstrate how a new nameserverset is entered in the system and a handle is returned.

Steps:

- 8.2.1.1 Define a nonexistent nameserverset, consisting of at least 2 and at most 5 nameservers and 1 contact handle.
- 8.2.1.2 Compose a CreateNsset EPP command
- 8.2.1.3 Send the CreateNsset command to the ENUM system
- 8.2.1.4 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The new nameserverset is stored in the database
 - The unique handle of the new nameserverset is returned

8.2.2 Update attributes of an existing nameserverset

Purpose: demonstrate how an existing nameserverset is updated in the system.

Steps:

- 8.2.2.1 Select the handle of an existent nameserverset from the database
- 8.2.2.2 Make some changes to the data: change one of the nameservers, add a nameserver (maximum is 5 nameservers per set), select another contact
- 8.2.2.3 Compose an UpdateNsset EPP command
- 8.2.2.4 Send the UpdateNsset command to the ENUM system
- 8.2.2.5 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The nameserverset is correctly updated in the database

8.2.3 Delete a nameserverset from the database

Purpose: demonstrate how a nameserverset is deleted from the system.

Steps:

- 8.2.3.1 Select the handle of an existent nameserverset from the database. The nameserverset may not be referenced in any number object.
- 8.2.3.2 Compose a DeleteNsset EPP command
- 8.2.3.3 Send the DeleteNsset command to the ENUM system
- 8.2.3.4 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The nameserverset is correctly deleted from the database

8.3 Numbers

8.3.1 Create number: new ENUM number delegation

Purpose: demonstrate how a new number delegation is entered in the system.

Steps:

- 8.3.1.1 Define a nonexistent telephone number. Make sure the telephone number does not start with any of the excluded prefixes.
- 8.3.1.2 Select an existing contact handle for this telephone number.
- 8.3.1.3 Create a valid token for this telephone number and contact.
- 8.3.1.4 Compose a CreateNumber EPP command including the token.
- 8.3.1.5 Send the CreateNumber command to the ENUM system
- 8.3.1.6 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The new number delegation is stored in the database
 - The telephone number of the number delegation is returned (format: +31101234567)
 - The create date and expiration date of the number delegation are returned

8.3.2 Update number: change delegation attributes

Purpose: demonstrate how changes can be made to a number delegation.

Steps:

- 8.3.2.1 Select an existent number delegation.
- 8.3.2.2 Changes the data: select a different nameserverset (handle) for this number delegation.
- 8.3.2.3 Compose an UpdateNumber EPP command.
- 8.3.2.4 Send the UpdateNumber command to the ENUM system
- 8.3.2.5 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The changed number delegation is stored in the database

8.3.3 Renew number: supply a new token for a delegation

Purpose: demonstrate how a number delegation can be renewed.

Steps:

- 8.3.3.1 Select an existent number delegation.
- 8.3.3.2 Create a valid token for this telephone number and contact.
- 8.3.3.3 Compose a RenewNumber EPP command including the token.
- 8.3.3.4 Send the RenewNumber command to the ENUM system
- 8.3.3.5 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - A new expiration date of the number delegation (adjusted to the expiry of the token) is stored in the database.

8.3.4 Transfer number: move the delegation from one registrar to another registrar

Purpose: demonstrate how a number delegation can be transferred (moved) to another registrar.

Steps:

- 8.3.4.1 Select an existent number delegation, related to another registrar.
- 8.3.4.2 Create a valid token for this telephone number and contact.
- 8.3.4.3 Compose a TransferNumber EPP command including the token.
- 8.3.4.4 Send the TransferNumber command to the ENUM system
- 8.3.4.5 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The telephone number of the number delegation is returned (format: +31101234567)
 - The create date and expiration date of the number delegation are returned

8.3.5 Delete number: remove the delegation from the database

Purpose: demonstrate how a number delegation is deleted from the system.

Steps:

- 8.3.5.1 Select an existent number delegation.
- 8.3.5.2 Compose a DeleteNumber EPP command
- 8.3.5.3 Send the DeleteNumber command to the ENUM system
- 8.3.5.4 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The number delegation is correctly deleted from the database

8.4 General

8.4.1 Info: ask for information on an object

Purpose: demonstrate how information about an object (number, contact or nameserverset) in the system can be retrieved.

Steps:

- 8.4.1.1 Select an existent number, contact or nameserverset.
- 8.4.1.2 Compose an Info EPP command: use the telephone number in case of an Info Number, use the handle in case of an Info Contact or Info Nsset.
- 8.4.1.3 Send the Info command to the ENUM system
- 8.4.1.4 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully

- In case of an Info Number:
 - All information concerning the number delegation (contact and nameserver handles, options) is returned;
 - All information about the validation (dates, method, validator) is returned;
 - All information about the token (tokendata: address, person, signature) is returned.
- In case of an Info Contact:
 - All information concerning the contact is returned: names, address, phone, etc;
 - The handle of the contact is returned.
- In case of an Info Nsset:
 - All hostnames are returned;
 - The handle of the nameserverset is returned.

8.4.2 Check whether a number is already in the database

Purpose: demonstrate how the existence of a number delegation can be checked in the system.

Steps:

- 8.4.2.1 Select an existent number delegation.
- 8.4.2.2 Compose a Check EPP command
- 8.4.2.3 Send the Check command to the ENUM system
- 8.4.2.4 Examine the server response and database content (if necessary). Expected results are:
 - The command is executed successfully
 - The existence of the number is confirmed.
 - A message like “less specific, more specific or exact number found” is displayed.

8.4.3 Poll for new messages for the registrar

Purpose: demonstrate how messages for a registrar can be polled and deleted from the system.

Steps:

- 8.4.3.1 Make use of a registrar account for which some transactions are executed (and therefore messages are waiting). This should be the case after a transfer performed by another party, or when tokens are (soon to be) expired.
- 8.4.3.2 Compose a Poll EPP command
- 8.4.3.3 Send the Poll command to the ENUM system
- 8.4.3.4 Examine the server response. Expected result:
 - The command is executed successfully.
 - The existence of messages is confirmed.
 - The date and details of the messages are displayed.
- 8.4.3.5 Compose a Poll command, including acknowledgement for deletion.
- 8.4.3.6 Send the Poll command to the ENUM system.
- 8.4.3.7 Examine the server response. Expected result:
 - The command is executed successfully.

9 Validation

9.1 The validation token

A validation token is a XML document format for conveying validation related information from validation entities to the registry. Its attributes and associated values contain information deemed to be necessary for asserting the right-to-use and revalidation.

The relevant parts of the validation token are signed by the VE using XML-Signature. This signature allows checking authenticity and origin of a token.

9.1.1 Validation token attributes

A token **MUST** contain the following attributes:

- A single validation "serial" string uniquely identifying a validation token for a certain VE.
- A single "e164number" attribute, containing the E.164 number in international format for which validation was carried out.
- A single "validator" id, identifying the VE.
- A single "method" id, identifying the method used by the VE for validation.
- A single "registrar" id, identifying the registrar for which validation was carried out.
- A single "createdate" attribute, containing the date of validation, formatted as "full-date" according to [RFC3339](#).
- A single "expiredate" attribute, containing the expiration date of the validation token, formatted as "full-date" according to [RFC3339](#).

A token **MAY** contain a "tokendata" section. The section contains information about the entity whose right-to-use is being asserted.

- A single "organisation" attribute, containing the full name of the entity.
- A single "commercialregisternumber" attribute, containing the entity's registration number.
- A single "title" attribute.
- A single "firstname" attribute.
- A single "lastname" attribute.
- A single "address" section, containing the following attributes:
 - A single mandatory "streetname" attribute
 - A single optional "streetnumber" attribute
 - A single optional "apartment" attribute
 - A single mandatory "postalcode" attribute
 - A single mandatory "city" attribute
 - A single optional "state" attribute
 - A single mandatory "country" attribute
- up to 10 "phone" attributes, containing full E.164 numbers
- up to 10 "fax" attributes, containing full E.164 numbers
- up to 10 "email" attributes.

9.1.2 Token signature

The validation token is generated by a validation entity and passed via a registrar to the registry which then acts upon the content of the token. A digital signature on the token guarantees that

- the token was indeed generated by the indicated VE (authenticity)
- the token was not tampered with in transit (integrity)
- auditing the validation process is possible (non-repudiation).

The cryptographic signature on the token follows XML-DSIG. As tokens are to be transmitted as part of an already XML based protocol (EPP), the transform as specified in “Exclusive XML Canonical Normalization” is used. In order to make the signature an integral part of the token the "enveloped"-signature mode is employed. The actual signature uses the RSA-SHA1 algorithm and relies on X.509 certificates.

9.2 Generic token verification process

Any token received by the registry must pass the following token verification process. Additional requirements to that generic verification may apply depending on the transaction request which contains the token. The transaction specific policy contains more details on this. To pass the generic token verification, a token must fulfill the following requirements:

1. The token must be syntactically correct („**parseable**“).
2. The token's **signature** must successfully **validate** against the included certificate.
3. The included certificate must be found among the registry's database of **active certificates**.
4. the certificate's usage limits must **permit** the **transaction** which contains the token.
5. The VE-ID associated to the certificate found in the registry database must match the **VE-ID** attribute of the **token**.
6. The **number** contained in the token must map to a kind of number known to the registry
7. The VE-ID associated to the certificate must be **allowed** to **create tokens** for the affected **kind of number**.
8. The token's **expiration timestamp** must not be in the past at the time it is being processed.
9. The token's **create timestamp** must not be after the **expiration timestamp**
10. The token's **validation serial** together with the VE-ID must be unique in the registry database

Possible error / warning conditions:

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000002	Error	Parse error
EN000038	Error	Token error - Not exactly one signature found
EN000039	Error	Token error – Wrong algorithm in CanoicalizationMethod
EN000040	Error	Token error – Wrong SignatureMethod used
EN000041	Error	Token error – Wrong Reference
EN000042	Error	Token error – URI attribute does not match ID
EN000043	Error	Token error – wrong algorithm in DigestMethod
EN000044	Error	Token error – not exactly one Transforms tag found
EN000045	Error	Token error – wrong Transformations
EN000046	Error	Token error – not exactly one certificate found
EN000047	Error	Token error – signature verification failed
EN000048	Error	Certificate not found in registry list
EN000049	Error	Certificate does not permit current transaction
EN000050	Error	Validation entity ID mismatch

<i>Condition ID</i>	<i>Condition kind</i>	<i>Condition name</i>
EN000010	Error	Unknown kind of number
EN000051	Error	Validation entity is not allowed for this kind of number
EN000052	Error	Token expired
EN000053	Error	Token not valid yet
EN000054	Error	Unable to parse timestamp
EN000055	Error	Duplicate validation serial
EN000057	Error	create date is after the expire date
EN000058	Error	token used outside of timeframe
EN000059	Error	Validation entity unknown

If the token does not pass all of the checks listed above, the transaction containing the token must not proceed. Otherwise, the transaction may proceed to the next policy check level.

10 Software toolkit

The ENUM toolkit is a collection of perl modules, serving as a client toolkit to interact with the registry. Additionally, it can be used to create and verify validation tokens. A set of sample programs provide command line tools for manual interaction with the registry.

The toolkit and the sample programs are not intended to serve as a full-fledged EPP-client-solution. The samples demonstrate the possible usage of the EPP commands and give an impression of how to construct or adapt your own programs.

Use of this toolkit is by no means mandatory. It's merely offered as a service to help registrars start up their connection to the registry.

10.1 Installation

The software can be acquired from www.enum.nl, and can be unpacked using standard tools. To install the package, perform the following steps:

- Install perl modules required by the package. The following perl modules are required by Number::E164:

```
Tree::Trie
YAML
```

Installation of those packages is out-of-scope here, those modules are available on CPAN.

- Download & unpack the archive, change into the installation directory:

```
$ wget [target URL]
$ tar -xzf Number-E164-0.01.tar.gz
$ cd Number-E164
```

- Prepare the Makefile for installation, make and test the package:

```
$ perl Makefile.PL
$ make
$ make test
```

- Install the package (you will need to become superuser for this step):

```
$ su
$ make install
```

10.2 Example installation details

This guide is an example installation on a virtual machine with Ubuntu 8.10.

```
# get the virtual machine used on vmplanet @ http://vmplanet.net/node/79
# the use of sudo is required for most commands
```

```
apt-get install libdate-calc-perl
apt-get install libio-socket-ssl-perl
apt-get install libwww-perl
apt-get install libxml-xerces-perl libxerces-c28 libxerces-c2-dev
apt-get install libssl-dev
apt-get install dh-make-perl
dh-make-perl --cpan Crypt::OpenSSL::X509 --build
```

```
# This creates an installer package
# (libcrypt-openssl-x509-perl_0.7-1_i386.deb)
# in the root of the filesystem
```

```
dpkg -i /libcrypt-openssl-x509-perl_0.7-1_i386.deb
apt-get install libxml-security-c14 libxml-security-c-dev
```

```
apt-file update
```

```
# extract the toolkit and enter the newly created toolkit directory
tar xvzf ENUMAT-ClientToolkit.tar.gz
cd ENUMAT-ClientToolkit
```

```
# build the toolkit
perl Makefile.PL
patch -p0 < xs-Makefile.patch
make
make test
make install
```

```
# Done
```

10.2.1 Testing - create a private key and certificate

The use of 'sudo' is required for most commands.

```
# go to the toolkit directory
cd cd ENUMAT-ClientToolkit

# create a directory for the toolkit
mkdir certificates

# create a private key and certificate:
cd certificates
openssl req -new -days 365 -x509 -out test.cert

# rename the key to test.key for testing purposes
mv privkey.pem test.key
```

10.2.2 Testing - create a token

This is an example of a create-number command which uses the token created above to register the geographic number '+ 31624101368'. The registrarid, validatorid, method and handles are unique for each registrar. The id's used in the examples must be changed to the correct id's.

The use of 'sudo' is required for most commands.

```
# go to the sample sub directory of the toolkit:
cd samples

# use the --keypass option if during creating the private key and
# certificate a password was given.
# in the example below, the assumption is that the key and certificate
# are in directory ../certificates

perl createtoken --tokenfile mytoken.xml --keypass maarten --keyfile
../certificates/test.key --certfile ../certificates/test.cert --serial 502
--e164number +31624101368 --validator 4001 --registrarid 9001 --method 555
--createdate 2009-02-06 --expiredate 2009-07-01

# this creates a signed token in file : mytoken.xml
# The token is required for the create-number, renew-number and
# transfer-number commands

perl createnumber --server 9001:passwordhere@eto.enum.nl:700 --e164number
+31624101368 --nsset N00002348-ENUMNL --tokenfile mytoken.xml

#Done
```

10.2.3 Sample-directory

The „sample“ directory contains a few sample programs, including command line wrapper for the set of transactions supported by the registry.

The toolkit and these sample programs are not meant to serve as a full-fledged EPP-client-solution. The samples merely demonstrate the possible usage of the EPP commands and give an impression of how to construct or adapt your own programs.

The samples are not installed by default with the toolkit, but need to be installed manually (eg. by copying the sample programs to „/usr/local/bin/“).

Sample programs:

```
checknumber
checktoken
createcontact
creatensset
createnumber
createtoken
deletecontact
deletensset
deletenumber
infocontact
infonsset
infonumber
poll
renewnumber
sample.pl
transfERNUMBER
updatecontact
updatensset
updatenumber
```