



# ENUM registrar manual

version: 1.0  
date: 25.05.2007  
(c) 2007 IENUM Ltd.

change history:

| <i>Date/version</i> | <i>Changes</i>           | <i>By whom</i> |
|---------------------|--------------------------|----------------|
| 25.05.07            | Initial document version | Axelm          |

## Table of Contents

|  |    |
|--|----|
| 1 Overview.....                              | 4  |
| 1.1 Definitions.....                         | 4  |
| 1.2 Role model.....                          | 4  |
| 1.2.1 Registry.....                          | 4  |
| 1.2.2 Registrar.....                         | 4  |
| 1.2.3 Validation Entity.....                 | 4  |
| 1.2.4 Registrant (Numberholder).....         | 5  |
| 2 Administrative procedures.....             | 5  |
| 2.1 Registrar signup.....                    | 5  |
| 2.2 Validation entity signup.....            | 5  |
| 2.3 Billing.....                             | 5  |
| 3 Supported number ranges.....               | 5  |
| 3.1 Geographic numbers.....                  | 6  |
| 3.1.1 Number definition and usage.....       | 6  |
| 3.1.2 ENUM registration guidelines.....      | 6  |
| 3.1.3 Example.....                           | 6  |
| 3.2 Mobile numbers.....                      | 6  |
| 3.2.1 Number definition and usage.....       | 6  |
| 3.2.2 ENUM registration guidelines.....      | 6  |
| 3.2.3 Example.....                           | 6  |
| 3.3 Personal Number Services.....            | 7  |
| 3.3.1 Definition and usage.....              | 7  |
| 3.3.2 ENUM delegation guidelines.....        | 7  |
| 3.3.3 Example.....                           | 7  |
| 3.4 VoIP Numbers.....                        | 7  |
| 3.4.1 Number definition and usage.....       | 7  |
| 3.4.2 ENUM delegation guidelines.....        | 7  |
| 3.4.3 Example.....                           | 7  |
| 4 Extensible Provisioning Protocol EPP.....  | 7  |
| 4.1 EPP overview.....                        | 8  |
| 4.1.1 Registry production server.....        | 8  |
| 4.1.2 Registry test server.....              | 8  |
| 5 Registry objects and transactions.....     | 8  |
| 5.1 Registry object types.....               | 8  |
| 5.1.1 Object relation.....                   | 9  |
| 5.1.2 Object ownership and access.....       | 9  |
| 5.1.3 Number type object.....                | 9  |
| 5.1.4 Contact type object.....               | 12 |
| 5.1.5 Nameserverset type object.....         | 16 |
| 5.1.6 Token type object.....                 | 17 |
| 5.2 Transactions (transform commands).....   | 21 |
| 5.2.1 create number.....                     | 21 |
| 5.2.2 update number.....                     | 24 |
| 5.2.3 delete number.....                     | 26 |
| 5.2.4 renew number (supply a new token)..... | 28 |
| 5.2.5 create contact.....                    | 30 |
| 5.2.6 update contact.....                    | 31 |
| 5.2.7 delete contact.....                    | 34 |
| 5.2.8 Create nameserverset.....              | 35 |

---

|  |    |
|--|----|
| 5.2.9 Update nameserverset.....              | 36 |
| 5.2.10 Delete nameserverset.....             | 38 |
| 5.2.11 transfer number.....                  | 40 |
| 5.3 Query commands.....                      | 42 |
| 5.3.1 „info“ command.....                    | 43 |
| 5.3.2 „check“ command.....                   | 44 |
| 5.3.3 „hello“ command.....                   | 45 |
| 5.3.4 „poll“ command.....                    | 45 |
| 5.4 EPP response extension.....              | 46 |
| 5.4.1 condition description.....             | 46 |
| 5.4.2 response example.....                  | 47 |
| 5.5 EPP message queue.....                   | 47 |
| 5.5.1 Message format.....                    | 47 |
| 5.5.2 Queue policy.....                      | 48 |
| 5.5.3 Message type definitions.....          | 48 |
| 6 Example Transactions.....                  | 50 |
| 6.1 EPP login.....                           | 50 |
| 6.2 Initial object creation.....             | 50 |
| 6.3 ENUM Domain for a geographic number..... | 50 |
| 7 Validation.....                            | 51 |
| 7.1 The validation token.....                | 51 |
| 7.1.1 Validation token attributes.....       | 51 |
| 7.1.2 Token signature.....                   | 52 |
| 7.2 Generic token verification process.....  | 52 |
| 7.3 Validation Token and Number Ranges.....  | 53 |
| 8 ENUMIE software Toolkits.....              | 54 |
| 8.1 ENUMIE toolkit.....                      | 54 |
| 8.1.1 Installation.....                      | 54 |

# 1 Overview

## 1.1 Definitions

- **PBX:** „private branch extension“ -
- **NDC:** „national destination code“ (sometimes called „area code“)
- **VE:** „Validation Entity“

## 1.2 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. In the case of the Irish ENUM registry system, Validation Entities are currently combined with the Registrar role, since experience in other countries has shown that there doesn't currently seem to exist a business case for a standalone Validation Entity.

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

### 1.2.1 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 Ireland, this role has been contracted to **IENUM Ltd**.

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 3.5.3.e164.arpa zone with NS records delegating the ENUM domain for individual numbers to other nameservers.

However, in the case of Ireland, IENUM Ltd also operates a Tier 2 portal as a „last resort“ registrar. This means that IENUM currently assumes the role of both the registry and a registrar.

### 1.2.2 Registrar

A Registrar requests ENUM domain delegations at the Registry on behalf of a numberholder. This is the same role Registrars fulfill in the ccTLD and gTLD world. Direct registration from a numberholder is not permitted. All delegation requests must be channeled through a Registrar.

### 1.2.3 Validation Entity

The Validation Entity (VE) is the party which confirms that the requesting numberholder 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 those tokens against preconfigured certificates. The tokens document that the VE has checked whether a prospective ENUM domain owner has the right to use on 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 (However, those roles are in most cases assumed by a single entity).

Validation Tokens link the following information together:

1. E.164 Number (or number range)
2. Registrar
3. Validation Entity
4. Validation Method
5. Validity timeframe
6. Numberholder Identity (optional)

Please note that such a token gives the registrar control over the ENUM domain of the numberholder. As the numberholder 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 numberholder. It is thus essential that the Validation Entity not only checks whether the numberholder has indeed the rights to the E.164 number, but also that he intends to register the corresponding ENUM domain with this specific registrar.

### 1.2.4 Registrant (Numberholder)

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 typically arises from an assignment by a telco (communication service provider – CSP) to a customer.

## 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 IENUM
- Sign the registrar contract, receive registry account data from **IENUM**
- Set up communication/provisioning 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). Typically, a Registrar will operate its own Validation Entity

### 2.2 Validation entity signup

Signing up with the the registry as Validation Entity involves the following steps (Please note that there is no separate contract – Registrars are automatically Validation Entities, too):

- Provide the registry with a list of intended validation methods, and keep this list up-to-date
- 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.

### 2.3 Billing

For billing purposes, the number of delegated ENUM domains per registrar at the end of each month is considered. More details regarding pricing can be found in the registrar contract.

## 3 Supported number ranges

The „Irish Numbering Plan“ as released by Commission for Communications Regulation on [www.comreg.ie](http://www.comreg.ie) is the basis for the following section. Ireland’s Numbering conventions are defined by

Comreg in the document “National Numbering Conventions” Comreg 05/62. This serves as the basis for the number ranges supported in ENUM. Specifically section 11.6 defines those ranges which are suitable for use within ENUM. This document is available at [http://www.comreg.ie/\\_fileupload/publications/ComReg0562.pdf](http://www.comreg.ie/_fileupload/publications/ComReg0562.pdf)

ENUM domains can not be assigned for number ranges which are not indicated below.

## **3.1 Geographic numbers**

### **3.1.1 Number definition and usage**

Those numbers consist of a variable length national destination code (length between 1 and 3 digits), and a fixed length subscriber number depending on the national destination code. Geographic number ranges are marked as „**local network code**“ in the numbering plan. National destination codes for geographic numbers currently start with digits 1 – 7 and 9, 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 only allowed, if the whole numberblock is assigned to one subscriber.

### **3.1.3 Example**

- **Original Number:** „+353 1 2365400“, where „+353“ indicates the country code for Ireland, „1“ indicates a geographic number in Dublin, „2365400“ indicates the subscriber number.
- **Number to be provisioned:** „+353 1 2365400“, 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 and related equipment. Numbers consist of a 2-digit NDC, and a 8 digits fixed length subscriber numbers.

### **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 „VPN-type“ 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:** „+353 83 12345678“, where „+353“ indicates the country code for Ireland, „83“ indicates the national destination code, and „12345678“ indicates the subscriber number.
- **Number to be provisioned:** „+353 83 12345678“

## 3.3 Personal Number Services

### 3.3.1 Definition and usage

Personal Numbering services allow the called person to receive calls at various different locations or terminals, including a mobile telephone, depending on the time the call is made or depending on other variables pre-defined by the called party (e.g. the location of and telecommunication facilities available to the called party at the time of the call).

Personal Number Services start with 3-digit NDC of „700“ and a fixed-length 6-digit subscriber number.

### 3.3.2 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.

### 3.3.3 Example

- **Original number:** „+353 700 123456“, where „+353“ indicates the country code for Ireland, „700“ indicates the NDC of personal number services, and „123456“ indicates the subscriber number.
- **Number to be provisioned:** „+353 700 123456”

## 3.4 VoIP Numbers

### 3.4.1 Number definition and usage

‘076’ VoIP telephone numbers are defined in the Comreg document ‘VoIP services in Ireland’ 04/103 (<http://www.comreg.ie/fileupload/publications/ComReg04103.pdf>).

Numbers in this range consist of a 2 digit NDC of „76“ and a 7 digit subscriber number.

### 3.4.2 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.

### 3.4.3 Example

- **Original number :** „+353 76 1234567“, where „+353“ indicates „Ireland“, „76“ indicates a VoIP number and „1234567“ indicates the subscriber number.
- **Number to be provisioned:** „+353 76 1234567”

## 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 evaluation by IENUM showed that the predefined objects of „vanilla“ EPP are not sufficient to handle the needs of an ENUM registry. To keep the protocol clean we 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 **registry.enum.ie** 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> IENUM test </svID>
    <svDate>2004-10-18T10:57:18.31Z</svDate>
    <svcMenu>
      <version>1.0</version>
      <lang>en</lang>
      <lang>de</lang>
      <objURI>http://www.enum.ie/rxsd/enum-number-1.0</objURI>
      <objURI>http://www.enum.ie/rxsd/enum-contact-1.0</objURI>
      <objURI>http://www.enum.ie/rxsd/enum-nsset-1.0</objURI>
      <objURI>http://www.enum.ie/rxsd/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 be 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 3.5.3.e164.arpa zone will not happen in realtime. The nameservers will reflect any update within about 20 minutes. Due to the caching behaviour of the DNS full propagation may take significantly longer.

### 4.1.2 Registry test server

An EPP test server will be available at **testregistry.enum.ie**, port **700**.

## 5 Registry objects and transactions

### 5.1 Registry object types

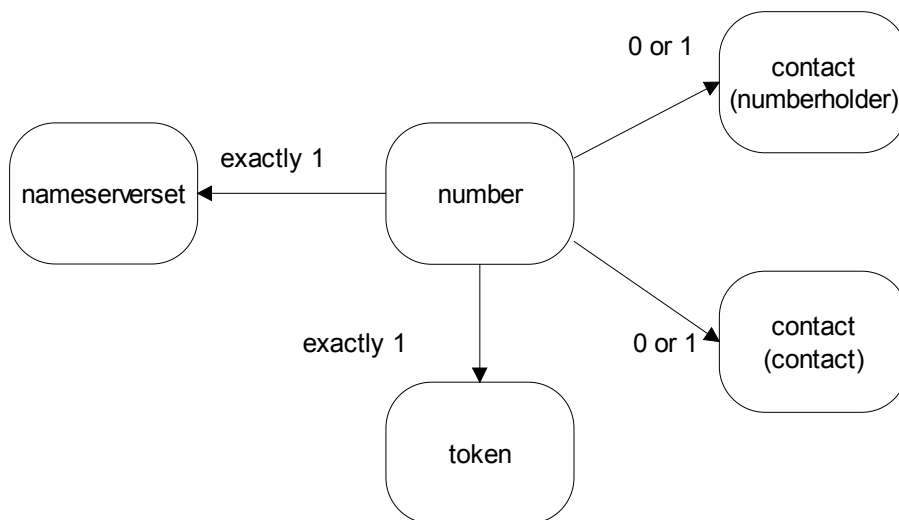
The registry supports the following object types:

- number

- contact
- nameserverset
- token

### 5.1.1 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 from several number or nameserverset type objects
- A nameserverset object may be referenced from several number type objects
- A token object for a certain number must only be referenced by a number object for which the token object is valid.

### 5.1.2 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 a „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**“.

### 5.1.3 Number type object

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

#### 5.1.3.1 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 |

| <i>Attribute name</i> | <i>type</i>   | <i>Cardinality</i> | <i>Description / Comment</i>   |
|-----------------------|---|--------------------|--|
| Numberholder          | Contact roid (type == person or type == organisation) | 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             | Contains NRH id as assigned by NUMA (not required for most number types)   |
| Whitepages            | Flag  | 1                  | Controls if number is to be listed in white pages directory. May not apply to all types of numbers                                   |
| directory_assistance  | Flag  | 1                  | Controls if number is to be disclosed by directory assistance services. May not apply to all types of numbers                        |
| online_directory      | Flag  | 1                  | Controls if number and numberholder is to be disclosed in online directories.  |
| disabled              | Flag  | 0                  | Controls if a delegation is to be excluded from DNS. Must be set in specific conditions as explained in the registrar contract only. |

### 5.1.3.2 Attribute policy

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

- the **e164number** attribute must successfully be associated with a number type known to the registry.
- If given, the **numberholder** attribute must contain a roid (repository object id) 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 **numberrangeholder** attribute must contain a NRH-id which is found among a fixed list of NRHs. The list of valid NRH-ids is published by the NUMA. This attribute is optional for most number ranges.
- If given, the **whitepages** flag must be either „true“, „false“, „0“ or „1“. If not given, defaults to „true“.
- If given, the **directory\_assistance** flag must be either „true“, „false“, „0“ or „1“. If not given, defaults to „true“.
- If given, the **online\_directory** flag must be either „true“, „false“, „0“ or „1“. If not given, defaults to „true“.

Please note that the attributes „whitepages“, „directory\_assistance“ and „online\_directory“ are currently not used for ENUM delegations under 3.5.3.e164.arpa, but are retained for future expansion. However, those attributes need to be present in requests in order to adhere to the XML schema. All three attributes should currently be set to „false“.

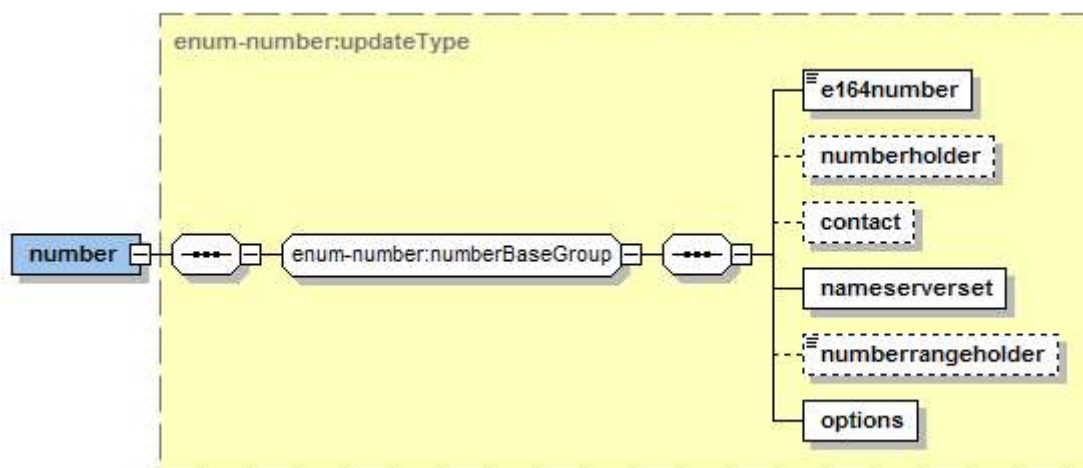
Possible error / warning conditions related to the attribute policy:

| <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.

### 5.1.3.3 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://www.enum.ie/rxsd/enum-number-1.0>“. A graphical representation of the schema follows:



### 5.1.3.4 Example

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

```
<number>
  <e164number>+353000000</e164number>
  <numberholder>C00012345-IENUM</numberholder>
  <contact>C00012346-IENUM</contact>
  <nameserverset>N00004711-IENUM</nameserverset>
  <numberrangeholder>22</numberrangeholder>
  <options whitepages="true" directory_assistance="true"
online_directory="true">
</number>
```

## 5.1.4 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.

### 5.1.4.1 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“<br/>subtype</i> | <i>„organization“<br/>subtype</i> | <i>„role“<br/>subtype</i> |
|----------------------------|--|-----------------------------|-----------------------------------|---------------------------|
| Roid                       | Contains the roid of the object.<br>Used to identify an contact object (primary key) | MUST                        | MUST                              | MUST                      |
| Type                       | Contains a string, describing the contact subtype                                    | „person“                    | „organisation“                    | „role“                    |
| Organisation               | Contains the full name of an organisation  | MAY                         | MUST                              | MAY                       |
| Commercial-register-number | Contains the organisation'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 „**organisation**“ subtype provides information about a legal entity, which is not a human person. The „**role**“ subtype describes an 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 number holder. 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 organisation as a whole in that case.

### 5.1.4.2 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.

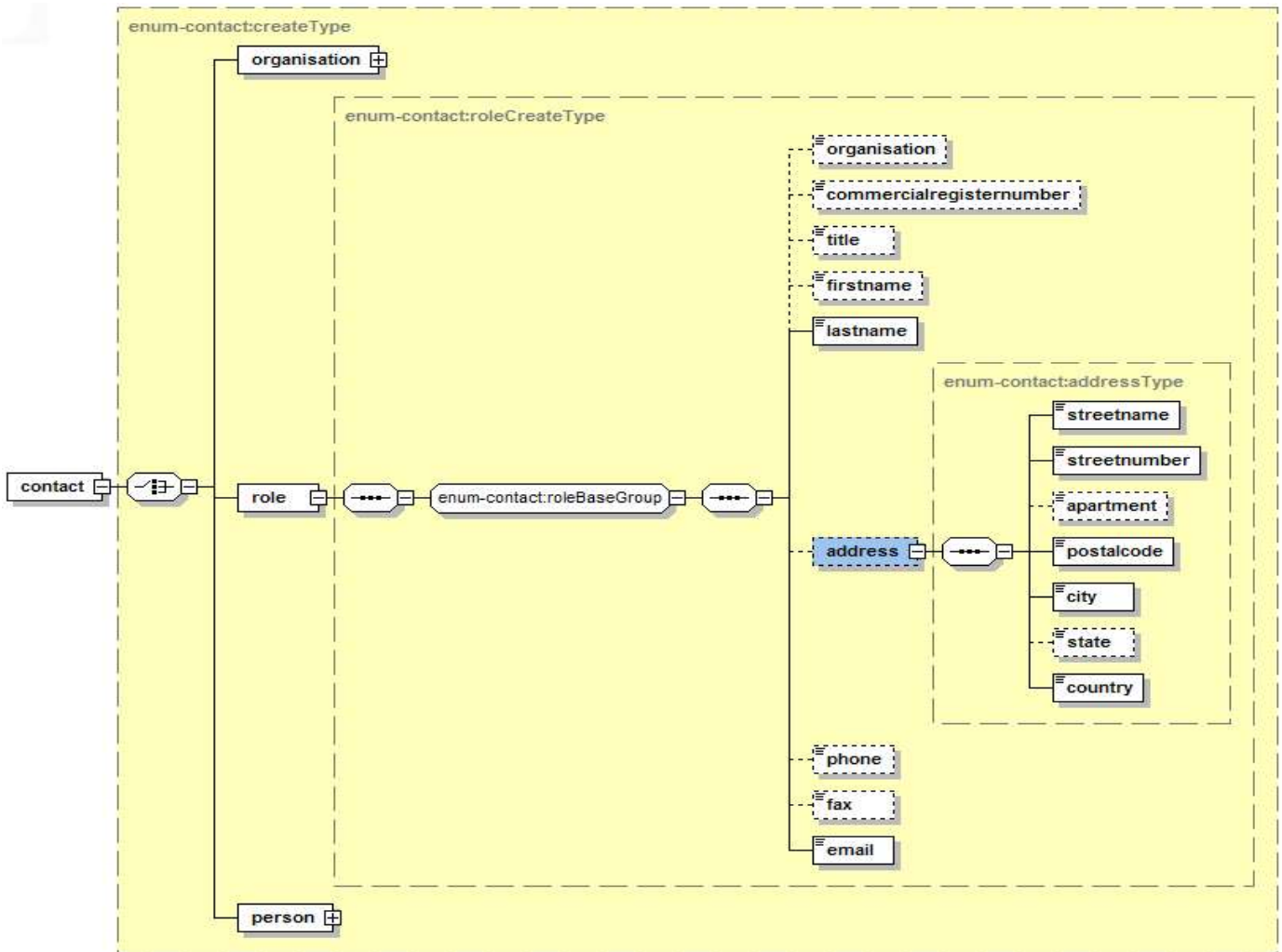
Possible error / warning conditions:

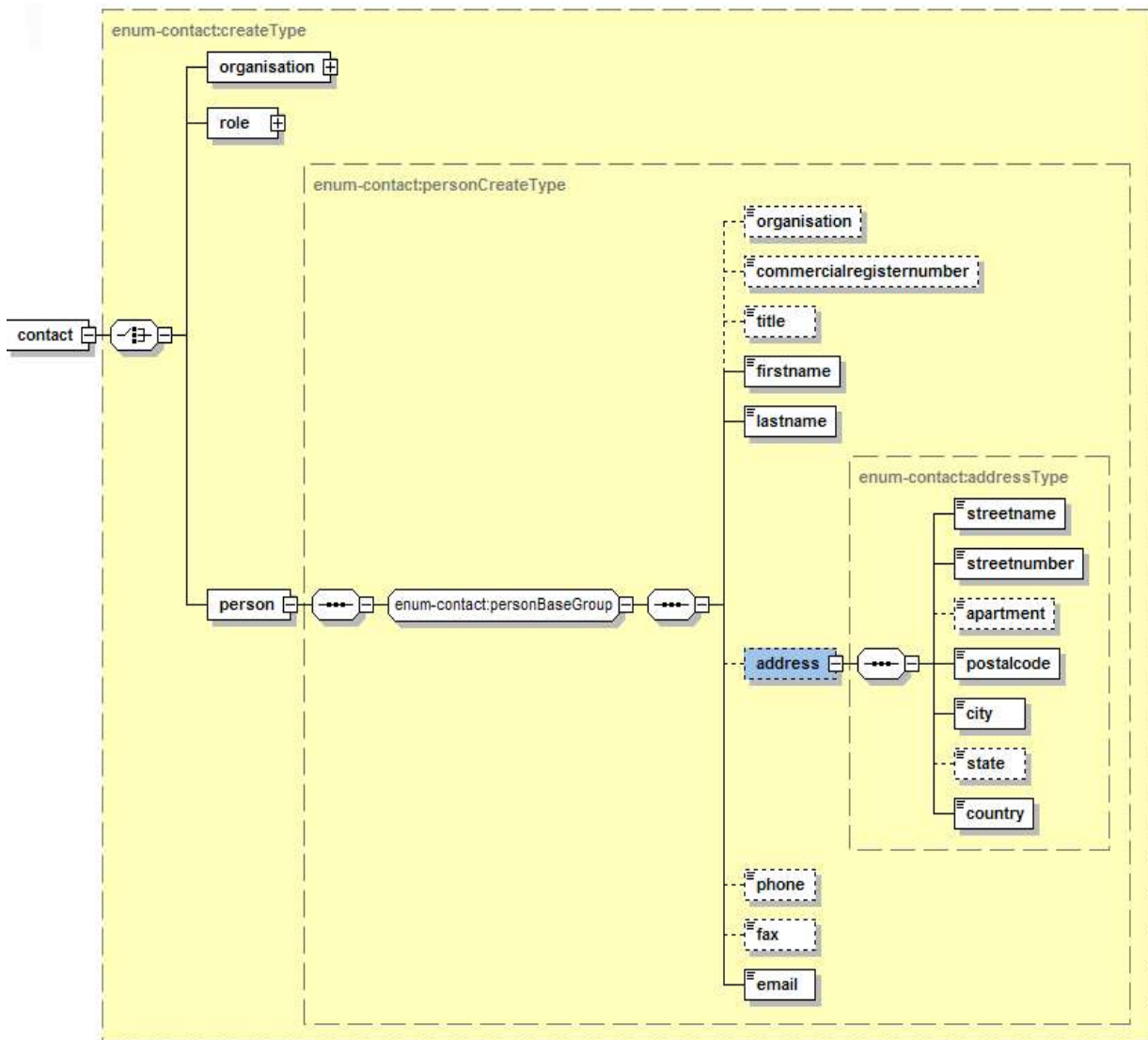
| <i>Condition ID</i> | <i>Condition kind</i> | <i>Condition name</i> |
|---------------------|-----------------------|-----------------------|
| EN000017            | Error                 | Unknown country       |
| EN000060            | Error                 | Invalid character     |

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

### 5.1.4.3 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://www.enum.ie/rxsd/enum-contact-1.0>“. A graphical representation of the schema for each contact type follows:





### 5.1.4.4 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-IENUM</roid>
  <title>Dr.</title>
  <firstname>Joe</firstname>
  <lastname>User</lastname>

  <address>
    <streetname>Windsor Terrace, Sandycove</streetname>
    <streetnumber>14</streetnumber>
    <city>Dublin</city>
    <country>IE</country>
  </address>
  <email disclose="false">office@ienum.ie</email>
</person>
```

### 5.1.5 Nameserverset type object

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

#### 5.1.5.1 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 full qualified host 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 full qualified hostnames of name servers, those nameservers will be the target of DNS delegations when the nameserverset is used in a number object. A minimum of 2 distinct nameservers are required, the maximum number of supported nameservers 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.

#### 5.1.5.2 Attribute policy

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

- the **roid** attribute must contain an 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“.

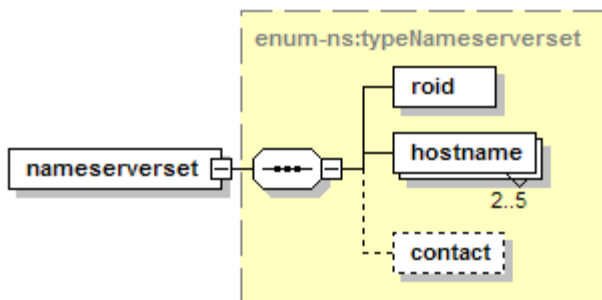
Possible error / warning conditions:

| <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))

### 5.1.5.3 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://www.enum.ie/rxsd/enum-nsset-1.0>“. A graphical representation of the schema follows:



### 5.1.5.4 Example

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

```
<nameserverset>
  <roid>N00012345-IENUM</roid>
  <hostname>ns01.enum.ie</hostname>
  <hostname>ns02.enum.ie</hostname>
  <hostname>ns03.enum.ie</hostname>
  <contact>C00004711-IENUM</contact>
</nameserverset>
```

### 5.1.6 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/internet-drafts/draft-ietf-enum-validation-token-03.txt>

### 5.1.6.1 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 it's validation tokens.                    |
| <E164Number>                | full qualified E.164 number for which validation was carried out. In case a „lastE164Number“ is provided, this attribute contains the first number of the range validated.      |
| <lastE164Number>            | Full qualified E.164 number indicating the last number of the E.164 number range for which that validation token is valid. Must be of same length as „E164Number“               |
| <validationEntityID>        | numeric (registry) ID of the VE   |
| <registrarID>               | numeric (registry) ID of the registrar  |
| <methodID>                  | numeric ID of the validation method used  |
| <executionDate>             | Date when the validation process was executed   |
| <expirationDate>            | 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. |

### 5.1.6.2 Attribute policy

- the **validationEntityID** must be known to the registry
- the **registrarID** must match the registrarid of the corresponding number object
- the **methodID** must be known to the registry, and must be valid for the kind of number affected.
- the **signature** must be valid.

### 5.1.6.3 Formal syntax (XML schema)

The validation token is defined in two separate schemas: The validation token core scheme (Namespace „urn:ietf:params:xml:ns:enum-token-1.0“ , File „enum-token-1.0.xsd“) and the token data scheme (Namespace: „urn:ietf:params:xml:ns:enum-tokendata-1.0“, File: „enum-tokendata-1.0.xsd“).

More details about the schema (including the schema itself) are contained in draft-ietf-enum-validation-token-03.

### 5.1.6.4 Example

This example was taken from the Internet Draft draft-ietf-enum-validation-token-03.txt – The signature of the token has been removed for legibility.

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<token xmlns="urn:ietf:params:xml:ns:enum-token-1.0" Id="TOKEN"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "urn:ietf:params:xml:ns:enum-token-1.0 enum-token-1.0.xsd">
  <validation serial="acmeve-000001">
    <E164Number>+442079460123</E164Number>
    <validationEntityID>ACME-VE</validationEntityID>
    <registrarID>reg-4711</registrarID>
    <methodID>42</methodID>
    <executionDate>2007-05-08</executionDate>
  </validation>
  <tokendata xmlns="urn:ietf:params:xml:ns:enum-tokendata-1.0"
    xsi:schemaLocation=
      "urn:ietf:params:xml:ns:enum-tokendata-1.0 enum-tokendata-1.0.xsd">
    <contact>
      <organisation>Example Inc.</organisation>
      <commercialregisternumber>4711</commercialregisternumber>
      <title>Dr.</title>
      <firstname>Max</firstname>
      <lastname>Mustermann</lastname>
      <address>
        <streetName>Main</streetName>
        <houseNumber>10</houseNumber>
        <postalCode>1010</postalCode>
        <locality>London</locality>
        <countyStateOrProvince>London</countyStateOrProvince>
        <ISOcountryCode>GB</ISOcountryCode>
      </address>
      <phone>+442079460123</phone>
      <email>mm@example.com</email>
    </contact>
  </tokendata>
  <Signature>
  <!-- signature removed - please see internet draft for full token -->
  </Signature>
</token>
```

## 5.2 Transactions (transform commands)

### 5.2.1 create number

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

#### 5.2.1.1 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** attribute of the number object to be registered must be covered by the **token** (number equal, or in case the token covers ranges, within the indicated range)
- 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.

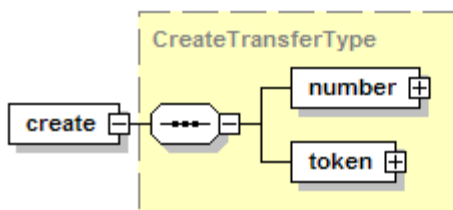
Possible error / warning conditions:

| <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     |

| <i>Condition ID</i> | <i>Condition kind</i> | <i>Condition name</i>  |
|---------------------|-----------------------|--|
| 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                                   |
| 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)

### 5.2.1.2 create number epp transaction schema



### 5.2.1.3 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://www.enum.ie/rxsd/enum-number-
1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0
          enum-number-1.0.xsd">
        <enum-number:e164number>+353123456789</enum-number:e164number>
        <enum-number:numberholder>C00012345-IENUM
          </enum-number:numberholder>
        <enum-number:contact>C00004711-IENUM</enum-number:contact>
        <enum-number:nameserverset>N00012312-IENUM
          </enum-number:nameserverset>
        <enum-number:numberrangeholder>22</enum-number:numberrangeholder>
        <enum-number:options whitepages="true" directory_assistance="true"
          online_directory="true" disabled="false" />
        <enum-token:token xmlns:enum-token="http://www.enum.ie/rxsd/enum-token-
1.0"
          xsi:schemaLocation="http://www.enum.ie/rxsd/enum-token-
1.0
            enum-token-1.0.xsd" ID="Signed">
        <enum-token:validation serial="23454">
          <enum-token:E164Number>+353123456789
            </enum-token:E164Number>
          <enum-token:validationEntityID>22</enum-
token:validationEntityID>
          <enum-token:registrarID>0815</enum-token:registrarID>
          <enum-token:executionDate>2004-01-01</enum-
token:executionDate>
  
```

```

                <enum-token:expirationDate>2005-01-01</enum-
token:expirationDate>
                </enum-token:validation>
                <enum-tokendata:tokendata
1.0"                xmlns:enum-tokendata="urn:ietf:params:xml:ns:enum-token-
1.0                xsi:schemaLocation="urn:ietf:params:xml:ns:enum-tokendata-
                enum-tokendata-1.0.xsd" >
                <enum-tokendata:contact>
                <enum-tokendata:firstname>Max</enum-tokendata:firstname>
                <enum-tokendata:lastname>Muster
                </enum-tokendata:lastname>
                </enum-tokendata:contact>
                </enum-tokendata:tokendata>
                <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
                <!-- signature removed -->
                </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"
  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-number:creData xmlns:enum-number
        xmlns:enum-number="http://www.enum.ie/rxsd/enum-number-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0
        enum-number-1.0.xsd">
        <enum-number:e164number>+353123456789</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>IENUM-200410141234</svTRID>
      </trID>
    </response>
  </epp>

```

## 5.2.2 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"> <li>• See transaction policy below</li> </ul>   |
|                | Locks:  | <ul style="list-style-type: none"> <li>• Write lock on the number.</li> <li>• Read lock on all referenced (new) contacts.</li> </ul> |
| Postconditions | <ul style="list-style-type: none"> <li>• The changes are made to the database.</li> <li>• The accounting data is modified.</li> <li>• Nameserver updates are being queued/triggered if necessary.</li> </ul>  |  |
| Remarks        | <ul style="list-style-type: none"> <li>• We propose <b>no</b> limits on what the registrar can do concerning nameserver, ownership and contacts of a domain under his control as long as the listed policy requirements are fulfilled.</li> <li>• We don't allow tokens in UPDATENUMBER. Tokens must be explicitly updated using RENEWNUMBER</li> </ul> |  |

### 5.2.2.1 Update number transaction policy

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

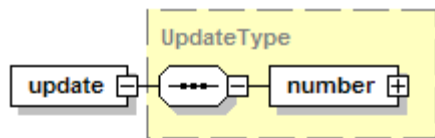
- The **client** must be allowed to provision transactions for the kind of number affected.
- The **e164number** attribute must describe an already existing object which is owned by the requesting registrar.
- The given **numberholder** must fulfill the numberholder policy of the affected kind of number, 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)

### 5.2.2.2 update number epp transaction



### 5.2.2.3 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://www.enum.ie/rxsd/enum-number-
1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0
        enum-number-1.0.xsd">
        <enum-number:number>
          <enum-number:options whitepages="true" directory_assistance="true"
online_directory="true" disabled="false">
            <enum-number:e164number>+3531427714277</enum-number:e164number>
            <enum-number:numberholder>C00223344-IENUM</enum-number:numberholder>
            <enum-number:contact>C00443322-IENUM</enum-number:contact>
            <enum-number:nameserverset>N00001122-IENUM</enum-
number:nameserverset>
            <enum-number:numberrangeholder>1234
              </enum-number:numberrangeholder>
            <enum-number:options whitepages="true" directory_assistance="true"
online_directory="true" />
          </enum-number:options>
        </enum-number:number>
      </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: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>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>IENUM-200410141254</svTRID>
    </trID>
  </response>
</epp>
```

### 5.2.3 delete number

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

**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“.

#### 5.2.3.1 Delete number transaction policy

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

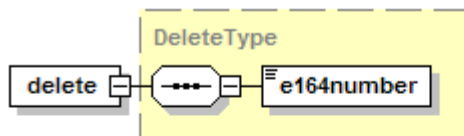
- The **registrar** must be allowed to perform a DELETENUMBER command for the affected kind of number.
- The **e164number** attribute must describe an object which is owned by the requesting registrar.

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  |
| 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)

#### 5.2.3.2 delete number epp transaction



### 5.2.3.3 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://www.enum.ie/rxsd/enum-number-
1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0
enum-number-1.0.xsd">
        <enum-number:e164number>+353427714277</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>IENUM-200410141266</svTRID>
    </trID>
  </response>
</epp>

```

## 5.2.4 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"> <li>See below.</li> </ul>           |
|                | Locks:   | <ul style="list-style-type: none"> <li>Read lock on number.</li> </ul> |
| Postconditions | <ul style="list-style-type: none"> <li>The new Token is added to the system, replacing the previous one</li> <li>The expiry attribute of the number object is adjusted to the expiry date of the new token.</li> </ul> |  |
| Remarks        | <ul style="list-style-type: none"> <li>A new Token replaces the previous one, regardless if the new token expires earlier than the old one.</li> </ul>   |  |

### 5.2.4.1 Renew number transaction policy

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

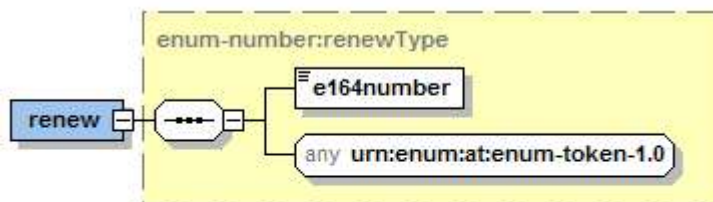
- The **registrar** must be allowed to perform transactions for the kind of number in question
- The **e164number** contained in the number object must refer to an existing number object owned by the requesting registrar.
- The **token** must pass the generic token verification
- The **e164number** contained to be renewed must be covered by the provided token.
- 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)

### 5.2.4.2 renew number epp transaction



### 5.2.4.3 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://www.enum.ie/rxsd/enum-number-
1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0
enum-number-1.0.xsd">
          <enum-number:e164number>+353123456789</enum-number:e164number>
          <enum-token:token xmlns:enum-token="urn:ietf:params:xml:ns:enum-
token-1.0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="urn:ietf:params:xml:ns:enum-token-1.0
enum-token-1.0.xsd" ID="Signed">
            <enum-token:validation serial="23454">
              <enum-token:e164number>+353123456789
              </enum-token:e164number>
              <enum-token:validationEntityID>12</enum-
token:validationEntityID>
              <enum-token:registrarID>0815</enum-token:registrarID>
              <enum-token:method>222</enum-token:method>
              <enum-token:executionDate>2004-01-01</enum-
token:executionDate>
              <enum-token:expirationDate>2005-01-01</enum-
token:expirationDate>
            </enum-token:validation>
            <enum-tokendata:tokendata
              xmlns:enum-tokendata="urn:ietf:params:xml:ns:enum-
tokendata-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:enum-tokendata-
1.0
                enum-tokendata-1.0.xsd" >
              <enum-tokendata:contact>
                <enum-tokendata:firstname>Max</enum-tokendata:firstname>
                <enum-tokendata:lastname>Muster
                </enum-tokendata:lastname>
              </enum-tokendata:contact>
            </enum-tokendata:tokendata>
            <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
              <!-- signature removed for this document -->
            </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"
  xmlns="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>IENUM-200410141288</svTRID>
    </trID>
  </response>
</epp>
```

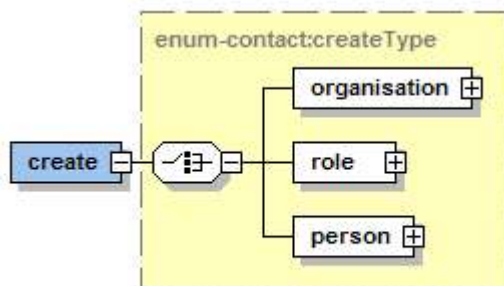
## 5.2.5 create contact

|                |   |  |
|----------------|---|--|
| Transaction:   | CREATECONTACT   |  |
| Description    | A contact type object is created  |  |
| Preconditions: | Attributes:   | <ul style="list-style-type: none"> <li>Contact information delivered in request must fulfill requirements as described in „contact object attribute policy“</li> </ul> |
|                | Policy:   | <ul style="list-style-type: none"> <li>No restrictions on contact creation.</li> </ul>   |
|                | Locks:  | <ul style="list-style-type: none"> <li>Write lock on object being created.</li> </ul>  |
| Postconditions | <ul style="list-style-type: none"> <li>A repository object id is generated</li> <li>The contact is added to the system</li> <li>The client receives the roid of the newly created contact</li> </ul>  |  |
| Remarks        | <ul style="list-style-type: none"> <li>roid namespace is shared between registrars, so that roids are unique to an instance of the system.</li> <li>roids are auto-generated by the registry – There's no way to provision a „personal handle“</li> </ul> |  |

### 5.2.5.1 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.

### 5.2.5.2 create contact epp transaction



### 5.2.5.3 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://www.enum.ie/rxsd/enum-
  contact-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/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"
    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-contact:creData xmlns:enum-contact
        xmlns:enum-contact="http://www.enum.ie/rxsd/enum-contact-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-contact-1.0
          enum-contact-1.0.xsd">
        <enum-contact:roid>C00012345-IENUM</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>IENUM-200410141234</svTRID>
    </trID>
  </response>
</epp>

```

**5.2.6 update contact**

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

### 5.2.6.1 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:

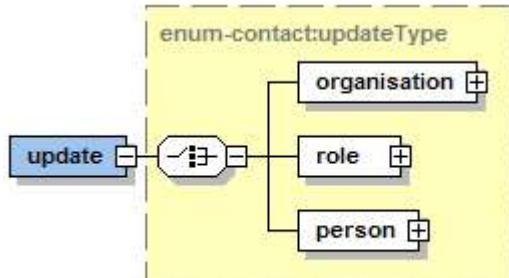
- The **roid** specified in the transaction request must describe an already existing contact which is owned by the requesting registrar.
- For any **number object** referencing the affected contact, the policy required by the kind of number of each affected number object must be fulfilled.
- The **type** 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)

### 5.2.6.2 update contact epp transaction schema



### 5.2.6.3 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://www.enum.ie/rxsd/enum-
contact-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-contact-1.0
enum-contact-1.0.xsd">
        <enum-contact:person>
          <enum-contact:roid>C00012345-IENUM</enum-contact:roid>
          <enum-contact:title>Dr.</enum-contact:title>
          <enum-contact:firstname>Joe</enum-contact:firstname>
        </enum-contact:person>
      </enum-contact:update>
    </update>
  </command>
</epp>
```

```
        <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"
    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>ENUMIE-200410141234</svTRID>
    </trID>
  </response>
</epp>
```

## 5.2.7 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"> <li>• The contact is deleted</li> <li>• Accounting log is written</li> </ul> |   |
| Remarks        |   |   |

### 5.2.7.1 Delete contact transaction policy

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

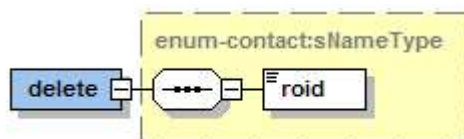
- The **roid** must describe an existing contact which is owned by the requesting registrar
- The **contact** must not be referenced in any number or nameserver object.

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)

### 5.2.7.2 delete contact epp transaction



### 5.2.7.3 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://www.enum.ie/rxsd/enum-
contact-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-contact-1.0
enum-contact-1.0.xsd">
```

```

        <enum-contact:roid>C00001234-IENUM</enum-contact:roid>
    </enum-contact: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"
    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>IENUM-200410141234</svTRID>
    </trID>
  </response>
</epp>

```

### 5.2.8 Create nameserverset

|                |   |   |
|----------------|---|---|
| Transaction:   | CREATENAMESERVERSET   |   |
| Description    | A nameserverset type object is created  |   |
| Preconditions: | nameserverset:  | <ul style="list-style-type: none"> <li>Request must fulfill nameserverset object attribute policies</li> </ul>                                |
|                | Policy:   | <ul style="list-style-type: none"> <li>No limits on creation of nameserverset</li> </ul>  |
|                | Locks:  | <ul style="list-style-type: none"> <li>Write lock on prospective nameserverset object, read lock on contact object, if referenced.</li> </ul> |
| Postconditions | <ul style="list-style-type: none"> <li>The nameserverset is created</li> <li>Accounting log is written</li> </ul>             |   |
| Remarks        | <ul style="list-style-type: none"> <li>Handles are auto-generated – Requesting a specific handle is not supported.</li> </ul> |   |

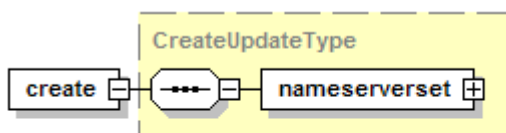
#### 5.2.8.1 Create nameserverset transaction policy

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.

#### 5.2.8.2 create nameserverset epp transaction



### 5.2.8.3 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://www.enum.ie/rxsd/enum-nsset-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-nsset-1.0
        enum-nsset-1.0.xsd">
        <enum-nsset:hostname>ns1.example.ie</enum-nsset:hostname>
        <enum-nsset:hostname>ns2.example.ie</enum-nsset:hostname>
        <enum-nsset:hostname>ns3.example.ie</enum-nsset:hostname>
        <enum-nsset:contact>C0001234-IENUM</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"
  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:creData
        xmlns:enum-nsset="http://www.enum.ie/rxsd/enum-nsset-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-nsset-1.0
        enum-nsset-1.0.xsd">
        <enum-contact:roid>N00012345-IENUM</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>IENUM-200410141334</svTRID>
    </trID>
  </response>
</epp>
```

### 5.2.9 Update nameserverset

|              |  |
|--------------|--|
| Transaction: | UPDATENAMESERVERSET                    |
| Description  | A nameserverset type object is updated |

|                |  |  |
|----------------|--|--|
| Preconditions: | nameserverset:   | <ul style="list-style-type: none"> <li>Request must fulfill nameserverset attribute policy requirements</li> <li>If contact is given, it must already exist, and be of contact type „person“ or „contact“</li> </ul> |
|                | Policy:  | <ul style="list-style-type: none"> <li>Handles of nameserverset must not be updated.</li> </ul>  |
|                | Locks:   | <ul style="list-style-type: none"> <li>Write lock on nameserverset object, read lock on contact object, if referenced.</li> <li>Read locks on all domains which reference the nameserverset</li> </ul>               |
| Postconditions | <ul style="list-style-type: none"> <li>The nameserverset is updated</li> <li>The nameserver updated are queued/triggered</li> <li>Accounting log is written</li> </ul>   |  |
| Remarks        | <ul style="list-style-type: none"> <li>An update to a nameserverset may trigger lot of nameserver updates. Those updates should be decoupled from the actual object update (nameserver update queue).</li> </ul> |  |

### 5.2.9.1 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:

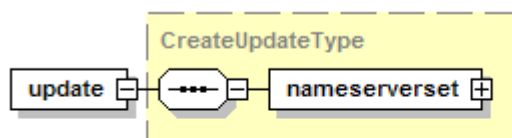
- the **registrar** must be allowed to perform UPDATENAMESERVERSET transactions.
- The **roid** must specify a nameserverset which is owned by the requesting registrar.

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)

### 5.2.9.2 Update nameserverset epp transaction



### 5.2.9.3 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://www.enum.ie/rxsd/enum-nsset-1.0"
      xsi:schemaLocation="http://www.enum.ie/rxsd/enum-nsset-1.0
        enum-nsset-1.0.xsd">
      <enum-nsset:roid>N1112223-IENUM</enum-nsset:roid>
      <enum-nsset:hostname>ns1.example.ie</enum-nsset:hostname>
      <enum-nsset:hostname>ns2.example.ie</enum-nsset:hostname>
      <enum-nsset:contact>C12340000-IENUM</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>IENUM-200410141234</svTRID>
    </trID>
  </response>
</epp>

```

## 5.2.10 Delete nameserver set

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

### 5.2.10.1 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:

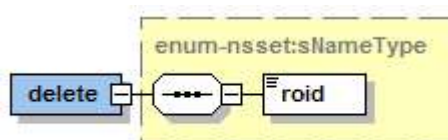
- The **roid** must specify a nameserver set which is not referenced from any number object, and which is owned by the requesting registrar.

Possible error / warning conditions:

| <b>Condition ID</b> | <b>Condition kind</b> | <b>Condition name</b>                   |
|---------------------|-----------------------|---|
| EN000014            | Error                 | Nameserverset does not exist            |
| EN000015            | Error                 | Nameserverset owned by different client |
| EN000034            | Error                 | Nameserverset is still in use           |

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

### 5.2.10.2 Delete nameserverset epp transaction



### 5.2.10.3 Delete nameserverset 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://www.enum.ie/rxsd/enum-nsset-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-nsset-1.0
          enum-nsset-1.0.xsd">
          <enum-nsset:roid>N00012345-IENUM</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>IENUM-200410141234</svTRID>
    </trID>
  </response>
</epp>
```

### 5.2.11transfer 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"> <li>Number must be a valid E.164 number</li> <li>Number must be already delegated to a different registrar.</li> <li>Number object must fulfill the number object attribute policies</li> </ul> |
|                | Token   | <ul style="list-style-type: none"> <li>The token must pass the generic token verification process</li> </ul>   |
|                | Locks:  | <ul style="list-style-type: none"> <li>Write lock on the number.</li> <li>Read lock on all referenced (new) contacts.</li> </ul>   |
| Postconditions | <ul style="list-style-type: none"> <li>The number is associated to the new registrar, and updated to reflect the new contact and nameserverset relations.</li> </ul>  |  |
| Remarks        | <ul style="list-style-type: none"> <li>We don't do complex operations. Person objects and nameserversets have to be created before they are referenced in a delegation request.</li> <li>The policy DB must be able to formulate rules based on the number, the registrar, and on the Validation Entity.</li> <li>De facto, this is a DELETENUMBER/CREATENUMBER sequence with another name (reason: different transaction allows for different policy)</li> <li>The losing registrar <b>MUST</b> be informed about a successful transfer by placing an message in his message queue.</li> </ul> |  |

#### 5.2.11.1Transfer 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 **number object** must be covered by the supplied validation token.
- 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

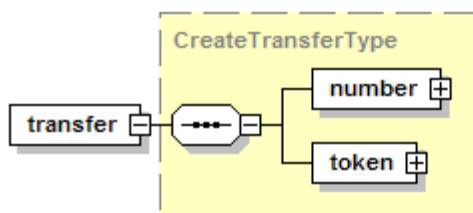
Possible error / warning conditions:

| <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 |

| <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                                  |
| 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)

### 5.2.11.2 Transfer number epp transaction



### 5.2.11.3 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://www.enum.ie/rxsd/enum-
number-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0
enum-number-1.0.xsd">
        <enum-number:e164number>+353123456789</enum-number:e164number>
        <enum-number:numberholder>C0012233-IENUM</enum-number:numberholder>
        <enum-number:contact>C00133324-IENUM</enum-number:contact>
        <enum-number:nameserverset>N00444334-IENUM
          </enum-number:nameserverset>
        <enum-number:numberrangeholder>2233
          </enum-number:numberrangeholder>
        <enum-number:options whitepages="true" directory_assistance="true"
online_directory="true" disabled="false" />
        <enum-token:token xmlns:enum-token="http://www.enum.ie/rxsd/enum-token-
1.0"
          xsi:schemaLocation="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.enum.ie/rxsd/enum-token-
enum-token-1.0.xsd" ID="Signed">
```

```

        <enum-token:validation serial="23454">
          <enum-token:e164number>+353123456789
            </enum-token:e164number>
          <enum-token:validationEntityID>234</enum-
token:validationEntityID>
          <enum-token:registrarID>0815</enum-token:registrarID>
          <enum-token:method>22</enum-token:method>
          <enum-token:executionDate>2004-01-01</enum-
token:executionDate>
          <enum-token:expirationDate>2005-01-01</enum-
token:expirationDate>
        </enum-token:validation>
        <enum-tokendata:tokendata
          xmlns:enum-tokendata="http://www.enum.ie/rxsd/enum-
tokendata-1.0"
          xsi:schemaLocation="http://www.enum.ie/rxsd/enum-
tokendata-1.0
            enum-tokendata-1.0.xsd" >
          <enum-tokendata:contact>
            <enum-tokendata:firstname>Max</enum-tokendata:firstname>
            <enum-tokendata:lastname>Muster
              </enum-tokendata:lastname>
            </enum-tokendata:contact>
          </enum-tokendata:tokendata>
          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
            <!-- signature removed for this document -->
          </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://www.enum.ie/rxsd/enum-number-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0
          enum-number-1.0.xsd">
        <enum-number:e164number>+353123456789</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>IENUM-200410151234</svTRID>
    </trID>
  </response>
</epp>

```

## 5.3 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:

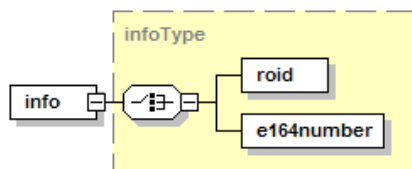
### 5.3.1 „info“ command

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.

#### 5.3.1.1 „info“ command formal syntax



#### 5.3.1.2 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://www.enum.ie/rxsd/enum-nsset-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-nsset-1.0
          enum-nsset-1.0.xsd">
        <enum-nsset:roid>N00012345-IENUM</enum-nsset:roid>
      </enum-nsset:info>
    </info>
    <c1TRID>ABC-12345</c1TRID>
  </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://www.enum.ie/rxsd/enum-nsset-1.0"
xsi:schemaLocation="http://www.enum.ie/rxsd/enum-nsset-1.0
enum-nsset-1.0.xsd">
  <enum-nsset:roid>N00012345-IENUM</enum-nsset:roid>
  <enum-nsset:status s="linked"/>
  <enum-nsset:status s="clientDeleteProhibited"/>
  <enum-nsset:hostname>ns1.example.ie</enum-nsset:hostname>
  <enum-nsset:hostname>ns2.example.ie</enum-nsset:hostname>
  <enum-nsset:contact>C00121212-IENUM</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>

```

### 5.3.2 „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“.

#### 5.3.2.1 „check number“ command example

The example checks for the number „+353 12345“ (fictitious 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://www.enum.ie/rxsd/enum-number-1.0"
xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0 enum-number-1.0.xsd">
<enum-number:e164number>+35312345</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://www.enum.ie/rxsd/enum-number-1.0"
xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0 enum-number-1.0.xsd">
  <cd>
    <e164number avail="0">+35312345</e164number>
    <reason>Less specific, more specific or exact number found</reason>
  </cd>
</chkData>
</resData>
<trID>
  <clTRID>111227238528633</clTRID>
  <svTRID>20050331123305498781-002-IENUM</svTRID>
</trID>
</response>
</epp>
```

### 5.3.3 „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.

### 5.3.4 „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.

#### 5.3.4.1 „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://www.enum.ie/rxsd/enum-message-1.0" type="number-
expired" xsi:schemaLocation="http://www.enum.ie/rxsd/enum-message-1.0 enum-message-
1.0.xsd">
        <numberList xmlns="http://www.enum.ie/rxsd/enum-number-1.0"
xsi:schemaLocation="http://www.enum.ie/rxsd/enum-number-1.0 enum-number-1.0.xsd">
          <number>
            <e164number>+353555</e164number>
            <exDate>2005-03-03T00:00:00.00Z</exDate>
          </number>
          <number>
```

```

        <e164number>+353999</e164number>
        <exDate>2005-01-30T00:00:00.00Z</exDate>
    </number>
</numberList>
</message>
</resData>
<trID>
    <clTRID>111227367825442</clTRID>
    <svTRID>20050331125437157204-002-IENUM</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-IENUM</svTRID>
        </trID>
    </response>
</epp>

```

## 5.4 EPP response extension

The IENUM 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.

### 5.4.1 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.

### 5.4.2 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="http://www.enum.ie/rxsd/enum-result-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/enum-result-1.0
        enum-result-1.0.xsd">
          <condition code='E0123' severity='fatal'>
            <msg>Person object 'IENUM-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>
          </condition>
        </conditions>
      </extension>
    </response>
  </epp>
```

---

```
    <details>The country 'IrelnD' as specified in the 'country'
      attribute is unknown to the registry, but is considered to be a
      frequent typo - it has been replaced by 'Ireland'</details>
  </condition>
</conditions>
</extension>
<trID>
  <clTRID>ABC-12345</clTRID>
  <svTRID>54321-XYZ</svTRID>
</trID>
</response>
</epp>
```

## 5.5 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.

### 5.5.1 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.

#### 5.5.1.1 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 IENUM message example</msg>
    </msgQ>
    <resData>
      <message
        xmlns="http://www.enum.ie/rxsd/enum-message-1.0"
        xsi:schemaLocation="http://www.enum.ie/rxsd/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>
```

### 5.5.2 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. “[IENUM 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:  
[IENUM registry] #4711 MT000001: number +3531234 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. “[registry@enum.ie](mailto:registry@enum.ie)”

- 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

### 5.5.3 Message type definitions

#### 5.5.3.1 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://www.enum.ie/rxsd/enum-message-1.0"
  xsi:schemaLocation="http://www.enum.ie/rxsd/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>
```

#### 5.5.3.2 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>+35312345678</e164number>
    <trDate>2004-12-22T09:00:00.0Z</trDate>
  </trnData>
</message>
```

#### 5.5.3.3 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>+3531234567</e164number>
      <exDate>2004-12-22T09:00:00.0Z</exDate>
    </number>
    <number>
      <e164number>+35398765543</e164number>
      <exDate>2004-12-22T12:32:33.4Z</exDate>
    </number>
  </numberList>
</message>
```

#### 5.5.3.4 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 containin 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”

#### 5.5.3.5 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”

#### 5.5.3.6 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”

## 6 Example Transactions

Here are some examples on typical business processes and how they will translate into transactions with the registry. We will use „example.com“ as the name and domain of the registrar.

### 6.1 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.

### 6.2 Initial object creation

All registrars will need some objects in the registry database which should be generated before any domains are delegated.

1. CREATE technical contact
2. CREATE nameserver set

### 6.3 ENUM Domain for a geographic number

This example assumes that the registrar will host the zones on his own nameservers. For simple geographic numbers, we skip the inclusion of a contact and numberholder objects for this domain.

1. Check if already delegated  
There are two ways of doing that: Either use the finger interface or the EPP info command.
2. Obtain Token  
How this is done depends a lot on the registrar himself. If he is identical to the telco who originally assigned the number to the customer, he will certainly act as VE himself. If not, the registrar can either make use of an external VE, or assume that role himself.  
Details on the Validation Token generation is out of scope of this document. See the Validation Entity handbook for further information.
3. Create Number / Transfer Number

## 6.4

# 7 Validation

### 7.1 The validation token

A validation token is a XML [6] 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.

The full specification for validation tokens is contained in [draft-ietf-enum-validation-token].

#### 7.1.1 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 "validationEntityID", identifying the VE.
- A single "methodID", identifying the method used by the VE for validation.
- A single "registrarID" id, identifying the registrar for which validation was carried out.
- A single "executionDate" attribute, containing the date of validation, formatted as "full-date" according to RFC3339.

A token **MAY** contain a single "expirationDate" attribute, marking the expiration date of the validation token, formatted as "full-date" according to RFC3339. If the expirationDate attribute is missing, the validation is considered to be valid infinitely. **Warning:** This may be limited to certain number ranges, e.g. Ranges where ENUM-Domain registration and number allocation are collapsed into a single step.

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.

Basically, all attributes are optional. In case an address section is used, several components are mandatory for conformance with the E.115 recommendation. The reason for this is that "computerized directory assistance" accessible through the E.115 interface may be a good source of validation information.

### **7.1.2 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.

## 7.2 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 **validationEntityID** 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** (expireDate) must not be in the past at the time it is being processed.
9. The token's **create timestamp** (executionDate) must not be after the **expiration timestamp**

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                            |
| 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                                |

| <i>Condition ID</i> | <i>Condition kind</i> | <i>Condition name</i>                |
|---------------------|-----------------------|--------------------------------------|
| 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 list'ed above, the transaction containing the token must not proceed. Otherwise, the transaction may proceed to the next policy check level.

### **7.3 Validation Token and Number Ranges**

Due to a closed numbering plan, enterprises etc. Requiring more than a a single number are allocated number blocks. Usually, those number blocks are „decadic“ (10, 100 or 1000 numbers), and can be provisioned in ENUM using a single delegation (shortened by 1, 2 or 3 numbers, covering the whole respective range).

However, communication service providers could also allocate numbers in non-decadic blocks (like +353 1234500 to +353 1234699).

Decadic as well as non-decadic blocks should be covered by a single validation token. The validation token contains two attributes to describe number ranges:

- e164number: Used to either describe a single number the token covers, or the first (lowest) number of the number range covered.
- LastE164Number: Used only in the case a number range/block is described.

In case a „lastE164Number“ is given, a validation token covers all ENUM domains which are entirely „contained“ in the indicated number range. Delegations may be shortened (optimized), so that the minimum number of delegations is needed for one number range.

#### **Examples**

- e164Number: +3531234500, lastE164Number: +3531234799: optimized delegations for „+35312345“, +35312346“, +35312347“
- e164Numer: +3531234824: single number, only +3531234824 to be provisioned
- e164Number: +3531234000, lastE164Number: +3531234999, can be covered with a single delegation for +3531234

## 8 ENUMIE software Toolkits

### 8.1 ENUMIE toolkit

The ENUMIE 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.

#### 8.1.1 Installation

The toolkit can be acquired from <http://www.enum.ie/>. Use tar/gz to unpack it. Before installing the toolkit itself, make sure that required packages are installed:

- Xerces, 2.3 preferred (newer versions may work as well, but are untested)
- Apache XML security library (<http://xml.apache.org/security/c/>)
- The following perl extensions:
  - Date::Calc
  - Crypt::OpenSSL:X509 (required OpenSSL)
  - XML::Xerces

Additionally, a working C++ build environment (compiler/linker) is required. After those prerequisites have been installed, the toolkit itself can be built and installed (superuser privileges are required for installation only):

```
$ perl Makefile.PL
[...]
```

```
$ make
[...]
```

```
$ make test
[...]
```

```
$ su
[...]
```

```
# make install
[...]
```

Note: The „sample“ directory contains a few sample programs, including command line wrapper for the set of transactions supported by the registry. Those sample products are not installed by default with the toolkit, but need to be installed manually (eg. by copying the sample programs to „usr/local/bin“).

## 8.2